DTIC FILE COPY

(2)

AD-A197 942

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DYNAMIC STALL CALCULATIONS USING A ZONAL NAVIER-STOKES MODEL

by

Jack H. Conroyd, Jr.

June 1988

Thesis Co-advisors: M.F. Platzer
Lawrence W. Carr

Approved for public release; distribution is unlimited

DTIC
ELECTE
SEP 0 2 1988
S E D

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) Code 31	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
DYNAMIC STALL COMPUTATIONS USING A ZONAL NAVIER-STOKES MODEL

12 PERSONAL AUTHOR(S)
Conroyd, Jack H. Jr.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year Month Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1988 June	212

16 SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Dynamic Stall; Navier-Stokes; Unsteady Aerodynamics

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A zonal Navier-Stokes model, developed by J.C. Wu, is installed and verified on the NASA Ames Cray X/MP-48 computer and is used to calculate the flow field about a NACA 0012 airfoil oscillating in pitch. Surface pressure distributions and integrated lift, pitching moment, and drag coefficient versus angle of attack are compared to existing experimental data for four cases and existing computational data for one case. These cases involve deep dynamic stall and fully detached flow at and below a freestream Mach number of .184. The flow field about the oscillating airfoil is investigated through the study of pressure, vorticity, local velocity and stream function. Finally, the effects of pitch rate on dynamic stall are investigated.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. M.F. Platzer	22b TELEPHONE (Include Area Code) (408) 646-2311; 22c OFFICE SYMBOL Code 67Pl

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

Approved for public release; distribution is unlimited

Dynamic Stall Computations Using a Zonal Navier-Stokes Model

by

Jack H. Conroyd, Jr.
Lieutenant, United States Navy, Reserve
B.S., Bradley University, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1988

Author: Jack H. Conroyd, Jr.
Jack H. Conroyd, Jr.

Approved by: M.F. Platzer
M.F. Platzer, Thesis Advisor

Lawrence W. Carr, Thesis Co-advisor

M.F. Platzer
M.F. Platzer, Chairman
Department of Aeronautics and Astronautics

G.E. Schacher
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

A zonal Navier-Stokes model, developed by J.C. Wu, is installed and verified on the NASA Ames Cray X/MP-48 computer and is used to calculate the flow field about a NACA 0012 airfoil oscillating in pitch. Surface pressure distributions and integrated lift, pitching moment, and drag coefficients and integrated lift, pitching moment, and drag coefficients versus angle of attack are compared to existing experimental data for four cases and existing computational data for one case. These cases involve deep dynamic stall and fully detached flow at and below a freestream Mach number of .184. The flow field about the oscillating airfoil is investigated through the study of pressure, vorticity, local velocity and stream function. Finally, the effects of pitch rate on dynamic stall are investigated.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification

By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 1

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Every endeavor is aided by those who provide guidance and assistance.

My sincere thanks to Larry Carr and Max Platzer for their patience, helpful suggestions and guidance. Their enthusiasm was contagious as they provided insight into various facets of fluid dynamics. To Patty Askling, who was the perfect, gracious host and allowed the extended use of her personal office space on many occasions. To all the others who extended their help and friendship along the way, it is all appreciated. And last, but certainly not least, to Greg Howe, whose humor, knowledge and unfailingly generous help were instrumental in the completion of this study. Thanks, Bud.

I. INTRODUCTION

Dynamic stall is a phenomenon that refers to an airfoil delaying stall beyond its static stall angle due to a rapid change in angle of attack. Associated with this is the generation of a strong vortex that appears at the leading edge of the airfoil which expands as it moves aft, causing large excursions in the pressure, pitching moment and lift the airfoil experiences (Figure 1, taken from [Ref. 1]).

Because the dynamic stall angle generally occurs at much higher angles of attack than the static stall angle, the maximum lift the airfoil generates can also be much higher than for steady conditions. Unfortunately this is a transient condition, as the lift drops sharply when the vortex is shed from the trailing edge. However, if the mechanisms that govern the initiation and development of this vortex and the dynamic stall phenomenon can be understood and controlled, it could open the door to much more maneuverable and higher performing aircraft. Nature provides a prime example of what is possible in the dragonfly, which uses vortex energy recovery to help achieve its remarkable maneuverability.

NASA currently has a flight test program utilizing vortex generating leading edge slats to gain further insight into this fascinating area. Parallel studies are underway

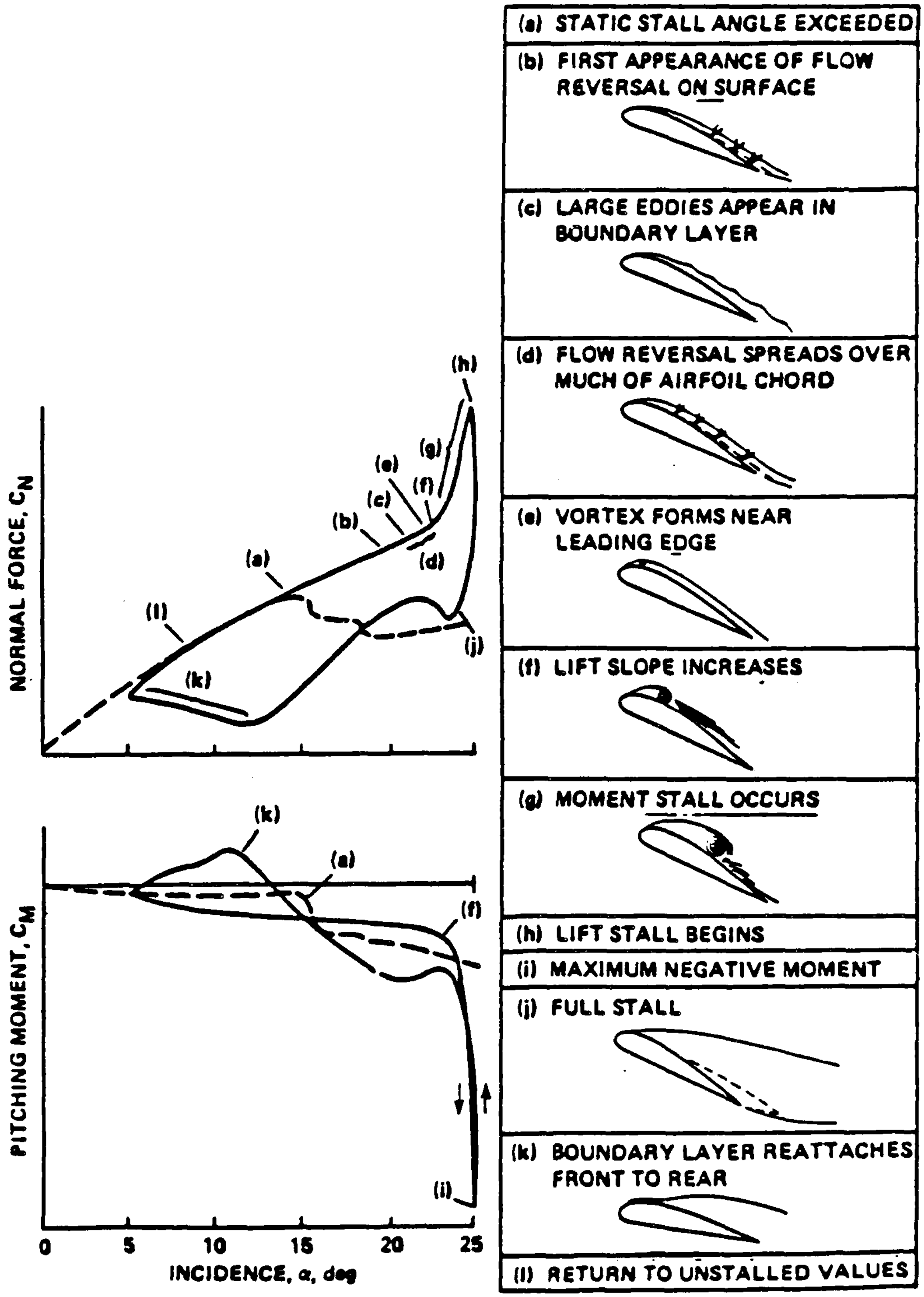


Figure 1. Dynamic Stall

by numerous groups to investigate dynamic stall via computational methods. The approaches that have received the most attention have used Navier-Stokes modeling, which has the flexibility to describe many flows.

Navier-Stokes modeling has usually been formulated along velocity/pressure lines. This formulation has two major problems [Ref. 2]. The first problem is caused by the size of the flow field, which in mathematical terms, is infinite in extent. Boundary conditions at infinity must be satisfied and an infinite flow field has to be modeled by a finite number of grid points. To do this, the boundary conditions must be previously known and multiple boundary condition solutions computed and compared to the known solution. Alternatively, a coordinate transformation can be made from the infinite flow field to a finite one. Unfortunately, this introduces a requirement for additional computations of an increasingly complex nature.

The second problem is the large number of data points required to adequately model the flow field. Even for two dimensional cases this can be a significant impediment. Various grid spacing methods have been used to concentrate data points in the regions around the airfoil where high flow variable gradients occur, but the number of data points required is still quite high.

An alternative method is to use velocity/vorticity modeling. By taking advantage of certain features of

velocity and vorticity fields for incompressible viscous flow, certain conclusions may be reached [Ref. 3].

1. Vorticity can be neither created nor destroyed in the interior of the fluid.

2. The total vorticity in the infinite unlimited space jointly occupied by the fluid and the solid is always zero.

3. The rate of convective transport of vorticity is finite. The rate of diffusive transport is effectively finite.

4. At large distances from the body, the velocity field approaches zero with increasing distance from the solid.

5. At large distances from the solid, the vorticity field decays exponentially with increasing distance from the solid.

6. The velocity field is uniquely determined by the vorticity distributed in the infinite unlimited space jointly occupied by the fluid and the solid. Alternatively, the velocity field is uniquely determined by the vorticity distribution in the fluid and the velocity condition on the solid boundary.

These features have three major salutary effects. [Ref. 4]

1. The actual number of grid points that need to be solved for will generally be much less than the total number of grid points needed to define the flow field. This is because vorticity is generated solely at the solid boundary, and this vorticity is diffused only a short distance from the solid before being carried away by convection and diffusion. Therefore, the flow will be inviscid a short distance ahead of the solid and often inviscid at small to moderate distances above and below the solid.

2. By taking advantage of the integral representation that the vorticity/velocity equations can be expressed in, the various regions of the flow field can be treated separately, with separate grids and distinct computational procedures that consider the length scales and defining equations, with no loss of accuracy and without a need to match the solutions of the various zones. Ultimately, an elegant method to compute flow field solutions becomes available that

can accurately compute dynamic stall conditions without resorting to brute force velocity/pressure finite difference computations.

3. The solution may be expressed as an integral which can be converted to a Fourier series expansion. [Ref. 5] This allows for very accurate computational solutions at each grid point.

Currently the model is limited to two-dimensional incompressible flows, but the concepts are applicable to three-dimensional and compressible flows as well.

A zonal Navier-Stokes model has been developed by J.C. Wu and his associates at the Georgia Institute of Technology and was made available for this study. An explicit, integro-differential methodology procedure is used to solve two-dimensional, Reynolds averaged, incompressible Navier-Stokes equations.

The goals of the present study were:

1. Install and verify the code on the Cray X/MP-48.

2. Compare the code's solutions with previous experimental and theoretical results.

3. Modify this code to enhance its utility.

II. MATHEMATICAL FORMULATION

A. GOVERNING EQUATIONS DEVELOPMENT

In the study of fluid flows, normally certain assumptions will be made. These assumptions allow emphasis on the specific features of interest, while providing useful simplifications in the theoretical development. Properly chosen, they can also promote the ease of numerical formulation and consequent solution. Accordingly, the assumptions made for the flow type in this study and their respective purposes are:

1. Two-dimensional. The primary benefit of this assumption is a very significant reduction in computational requirements. A relatively minor amount of information is lost for spanwise flow and tip effects, with the salient aspects to be investigated remaining intact.

2. Incompressible. A useful simplification to highlight the dynamic stall effects and allow for a more readily formulated computational model.

3. Viscous. A necessary element to describe the solid body and flow field interaction. This also helps to control the extent of flow field computations required.

4. Unsteady. Required to allow for airfoil pitching.

5. Turbulent. Required to describe the flow field under the conditions of interest.

6. Reynolds averaged. A formulation of the Navier-Stokes equations for turbulent conditions.

1. <u>The Navier-Stokes Equations</u>

For three-dimensional unsteady flow in the x-direction:

$$\rho \frac{Du}{Dt} = \rho X - \frac{1}{\rho}\frac{\partial p}{\partial x} + \mu \nabla^2 u$$

or
$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right)$$

$$= \rho X - \frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right]$$

where:

$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}$ = substantial derivative of u

ρX = body forces

$\frac{\partial p}{\partial x}$ = pressure forces

$\frac{\mu}{\rho}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right]$ = viscous term.

2. <u>Reynolds Averaging for Turbulent Flows</u>

The boundary layer equation of motion in two dimensions is [Ref. 6]:

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right)$$

letting the following variables be defined as:

$$u = U(x,y) + u'(x,y,t)$$

$$v = V(x,y) + v'(x,y,t)$$

$$p = P(x) + p'(x,y,t)$$

where:

$U(x,y)$ = mean x-direction velocity value over time

u' = fluctuation value of velocity in x-direction

$u' << U$

p = pressure

Then:

1. Substituting u, v and p into the boundary layer equation,
2. Taking the mean value of each term,
3. Observing that the mean value linear terms in a fluctuation component vanishes,
4. Assuming that the derivatives of the mean value linear term vanishes, and
5. Neglecting turbulent normal stress compared to the shearing stress,

gives:

$$\rho\left(U \frac{\partial U}{\partial x} + V \frac{\partial U}{\partial y}\right) = -\frac{dP}{dx} + \frac{\partial}{\partial y}\left(-\rho\overline{u'v'}\right)$$

where:

$-\rho\overline{u'v'}$ = the Reynolds stress.

Generally, the neglected terms are much smaller than $-\rho\overline{u'v'}$, and it should be noted that u' and v' can reach values as high as .1 U. However, for the flow under consideration, these are still valid approximations.

3. Prandtl's Mixing Length

From the Reynolds stress development, another concept can be presented [Refs. 7,8]. Boussinesq introduced a mixing coefficient that related the Reynolds stress to the derivative for U. This is possible because for there to be a net momentum transfer from the higher to lower momentum layers $-\rho\overline{u'v'} > 0$ or, in other words, u' and v' must be positively correlated. This is done by setting the Reynolds stress equal to the derivative of U or:

$$-\rho\overline{u'v'} = \nu_{\tau} \frac{dU}{dy}$$

where:

ν_{τ} = eddy viscosity which is a turbulent mixing coefficient.

Prandtl observed that ν_{τ} depends on U and is not a property of the fluid. A relation between ν_{τ} and the mean velocity was needed. Prandtl's idea was that if small "lumps" of fluid move from a lower to a higher average velocity location, the difference in its velocity compared to the surrounding mean velocity could be given by:

$$\Delta u \approx \ell\left(\frac{dU}{dy}\right)$$

where:

$U = f(y)$ for the simplified case,

ℓ = Prandtl's mixing length.

The physical significance of ℓ is the distance in the y direction the small lump of fluid must travel so that the difference between its velocity and U is equal to v'; which is the y-direction fluctuation velocity at that location. After further development, Prandtl observed that:

$$-\rho\overline{u'v'} = \rho\ell^2\left(\frac{\partial U}{\partial y}\right)^2 = \text{turbulent shearing stress}$$

This relation is very useful in the calculation of turbulent flows.

B. GOVERNING EQUATIONS

The equations of motion for the flow field under consideration are:

Velocity field: $\nabla \cdot \vec{v} = 0$ or $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$

Vorticity field: $\nabla \times \vec{v} = \vec{\omega}$ or $\left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right) = \vec{\omega}$

Navier-Stokes in the x-direction: $\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right]$

which, after taking the curl of both sides of the equation and applying the definitions of continuity and vorticity becomes [Ref. 9]:

$$\frac{\partial \omega}{\partial t} = (\omega \cdot \nabla)v - (v \cdot \nabla)\omega + \nu \nabla^2 \omega$$

where:

$(\omega \cdot \nabla)v$ = stretching and rotation

$(v \cdot \nabla)\omega$ = convection

$\nu \nabla^2 \omega$ = diffusion

It should be noted that for turbulent flows, ν should be replaced by ν_e, or:

$$\nu_e = \nu_\tau + \nu$$

where:

ν_e = effective viscosity

ν_τ = eddy viscosity

$\nu = \frac{\mu}{\rho}$ = kinematic viscosity.

C. ANALYTICAL FORMULATION

1. Kinematics

Kinematics is the branch of dynamics which deals with the motion of bodies without reference to the forces acting on the bodies. Of the three governing equations, the

two that comprise the kinematics of the flow are the equations of continuity and vorticity. Together, they express the relationship between velocity and vorticity throughout the fluid at any point in time.

There are two noteworthy aspects of these relations [Refs. 10,11]:

1. The differential equations are linear, hence they can readily be formulated for solution by computational methods.

2. The stress-strain relation is not a factor in these equations. This allows the fluid and solid to be treated together as one kinematic system which greatly simplifies the computations.

Through the use of fundamental solutions [Ref. 10] the continuity and vorticity field equations can be expressed as [Ref. 5]:

$$\vec{v}(\vec{r},t) = -\int_R \vec{\omega}_0 \times \nabla_0 P dR_0 + \oint_B [(\vec{v}_0 \cdot \vec{n}_0) - (\vec{v}_0 \times \vec{n}_0)] \times \nabla_0 P dB_0$$

where:

the subscript "0" refers to the $\vec{r}_0$ space,

$\vec{r}_0$ space is defined by the vorticity field, or, $\vec{\omega}_0 = \vec{\omega}(\vec{r}_0,t)$,

B is the boundary of the region R,

$\vec{n}$ is the outward unit normal vector on B, and

$P = -\frac{1}{2\pi} \ln \frac{1}{|\vec{r}-\vec{r}_0|}$ which is the fundamental solution for a two-dimensional Poisson equation.

The consequences of this formulation include:

1. The first integral will be zero for the inviscid region, therefore it needs to be computed only in the viscous region. This greatly reduces the

computational time required, as the viscous region is only a fraction of the total flow field.

2. The integrals can be expanded by Fourier series. This will provide more accurate solutions at each grid point than is possible via a straight finite difference method.

3. The attached viscous and detached viscous zone solutions may be computed separately. This allows an optimized grid spacing to be employed in each zone, where the respective length scales vary greatly. Figure 2 presents a flow zone portrayal.

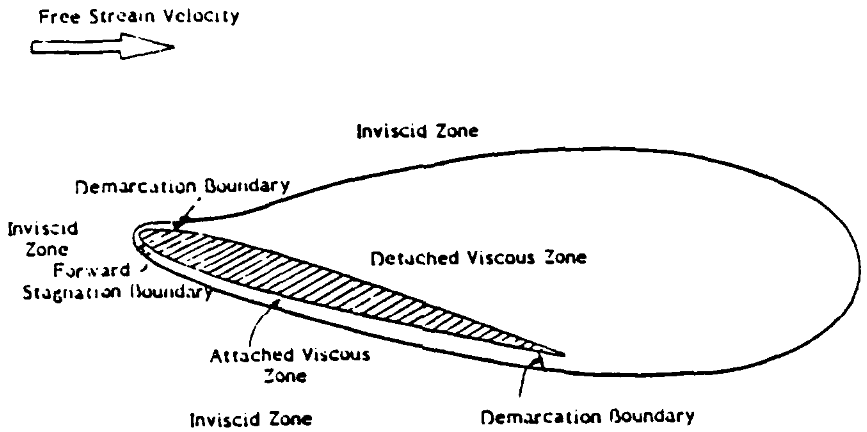


Figure 2. Flow Zones

4. Repeated computations and comparisons with either the boundary values or between the different flow zones are not required.

5. The question of modeling a strong viscous-inviscid interaction is obviated completely.

It should be noted that all of these aspects either reduce the number of calculations required, or enhance the accuracy of the solution. The net result is an elegant

solution to the conundrum encountered by those who wish to model this type of flow.

2. Kinetics

Kinetics is the branch of dynamics which deals with the effects of forces on the motion of bodies. The vorticity transport equation falls into the realm of kinetics. It is nonlinear and elliptic in space. This requires knowledge of the solution for the entire boundary conditions. For the outer boundary, this is met at the surface just inside the inviscid region. This boundary can change with each time step, but from the initial conditions, the vorticity will be zero along this surface. The interior boundary is the solid, and the vorticity values on this surface will need to be computed each time step. Because the vorticity values on the surface of the solid are not independent of the vorticity values in the interior of the fluid, this will have to be solved iteratively.

3. Grid Generation

In computational fluid dynamics, grid choice is something of an art. There are a number of conflicting goals to be considered. These include:

1. An adequate number of grid points to accurately represent the flow conditions.
2. A small enough number of grid points to help moderate the computational requirements.
3. Proper grid spacing demands including a fine mesh in the regions of high flow variable gradients and a coarser mesh in the other regions. The latter will help moderate the computational resources required.

4. A grid that is easy to generate.

5. A grid that simplifies and speeds computations.

Fortunately, due to the present mathematical formulation, a very efficient grid scheme may be employed.

By choosing the computational grid to be circular with radial lines, grid generation and computation goals can be met. Because the flow zones can be computed separately, grid spacing can be optimized for each zone. Due to the relatively small distances from the solid that the vortical flow is transported to, the size of the grid required can be moderated. The only drawback to this method is that the computational plane must be conformally mapped onto the physical plane. This limits the airfoil selection to those that can be accurately mapped between planes. Fortunately, a number of airfoils are available via Joukowski transforms, including the NACA 0012, for which there is a wealth of previous data available for comparison.

A salutary effect of using a Joukowski transform between the computational and physical planes is that the grid density in the physical plane is concentrated radially about the leading and trailing edges while becoming more sparse radially over the upper and lower surfaces of the airfoil.

Lastly, the computational grid is body fixed, which eliminates having to recalculate the grid for each time step. Representative grids are included in Figures 3-6.

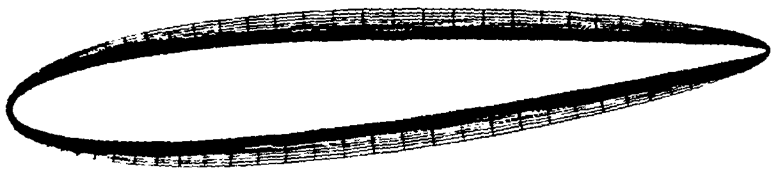


Figure 3. Boundary Layer Grid

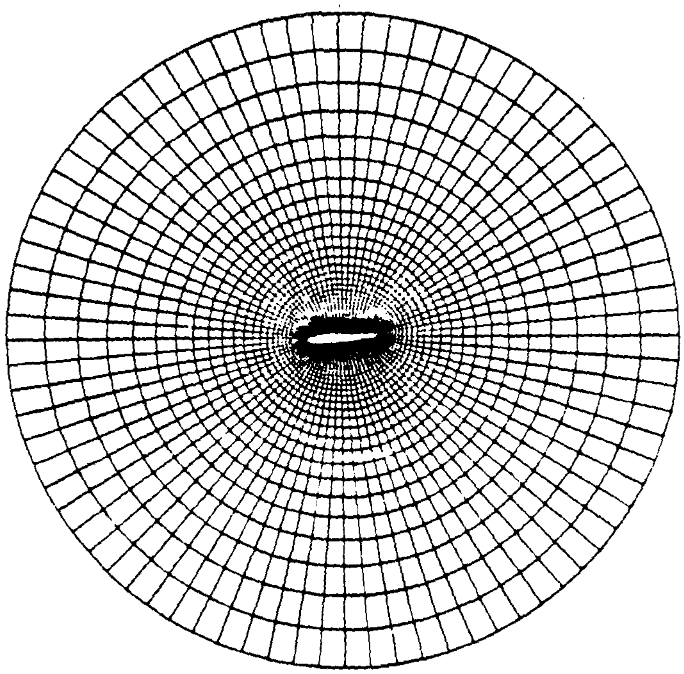


Figure 4. Vorticity Grid

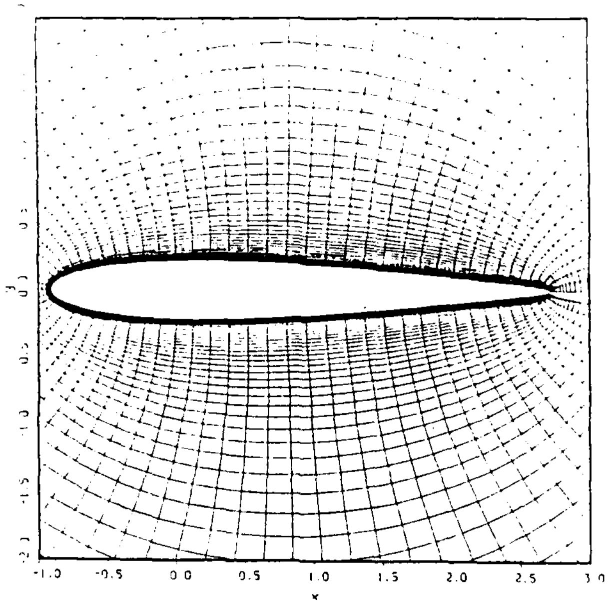


Figure 5. Near Field

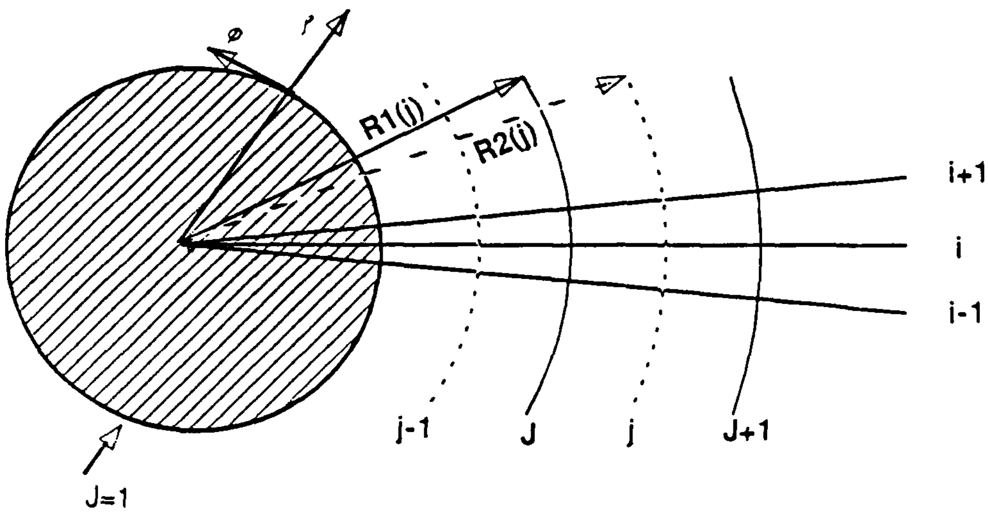


Figure 6. Computational Grid

4. Initial and Boundary Conditions

Initially, the solid and the fluid are at rest. The solid is then impulsively started, and the discontinuity between the solid and the fluid generates a vorticity sheet at the boundary. As time progresses, vorticity is diffused away from the solid and transported into the fluid by both diffusion and convection.

The four boundary conditions that must be met are for velocity and vorticity at the outer surface of the viscous region and at the solid/fluid boundary. As was previously mentioned, the outer vorticity boundary condition is met by setting the boundary just inside the inviscid region. The outer velocity boundary condition will simply be U.

The inner vorticity boundary condition, as previously stated, will need to be computed iteratively each time step, with the inner velocity boundary condition found by first determining the tangential velocity component, which is due to the solid's speed, angle of attack and oscillatory motion [Ref. 5] and then using that value and the relation $\vec{v} = \vec{v}_{solid}(\vec{r},t)$ [Ref. 13], to compute the inner velocity boundary condition.

III. DESCRIPTION OF THE CODE

The entire runstream has been named ZETA, which is an acronym for zonal procedure for evaluating turbulent and laminar flows. ZETA is composed of three primary sections:

1. GEOM--grid generation and transformation,
2. ZONST--main program in ZETA runstream, and
3. Plotting routines--consisting of pressure computations and various plotting options.

The computational loop in ZONST consists of [Ref. 5]:

1. Computing interior vorticity values by using the vorticity and velocity values from the previous time step in the vorticity transport equation.
2. Computing new boundary vorticity values.
3. Computing new velocity values.

Appendix B contains the version of the Wu code used for this study and Appendix C contains notes on its employment at NASA Ames.

A complete cycle through 40 degrees of pitch change at a reduced frequency of .15 requires 950 time steps and approximately 800 seconds of CPU time on the Cray X-MP/48 for an average of .842 seconds per time step. Initialization can be completed with 25 time steps. These are much more modest requirements than codes of similar ability.

A. GEOM

GEOM begins by reading the transformation and grid parameters. The transformation parameters define the

airfoil that will be used while the grid parameters define the size and stretching coefficients which control the radial grid spacing.

GEOM next generates the computational plane and then conformally maps it onto the physical plane. The outputs from GEOM are used by all the other programs. Grid point spacing may be checked here and adjusted as necessary.

B. ZONST

ZONST is the main program of ZETA. It uses the governing equations to compute the vorticity and velocity fields and generates the output used by the plotting routines.

It starts by reading the grid information from GEOM and its own input parameters. The input options available include:

1. Airfoil motion. This can be defined as a constant angle of attack, a rapidly pitching or an oscillating motion. All pitching airfoil flow solutions require data from the appropriate constant angle of attack steady state case to be stored as part of the initial conditions.

2. Time specifications allow the user to control the number of time steps to be run; the increment of the time steps and the time step values for which numerical and graphical output will be generated.

3. Flow zone specifications can be used to control the boundary layer region where Navier-Stokes computations will be used. Generally, there will be no increase in accuracy but a significant increase in computation time when using Navier-Stokes vice boundary layer calculations in the boundary layer region.

4. Under-relaxation parameters allow tailoring of the computational methods to help optimize computing efficiency for various flow conditions.

ZONST can be run for a specified period, checked and restarted. Should divergence occur, the last computed time step will be printed for trouble shooting.

The output from ZONST is numerical, and via plotting routines, graphical.

C. PLOTTING ROUTINES

Due to the primary variables used in ZONST, the aerodynamic loads caused by the pressure distribution are not directly available. LOADS uses the vorticity information and Fourier series expansions of the vorticity integral representation to derive values for coefficients of pressure, lift, drag and moment for each angle of attack.

The first three plotting routines listed use DISSPLA for their operating software, while the last one uses PLOT3D.

Plot1 portrays the physical plane vorticity grid, the boundary layer grid, flow zone demarcations and wake and turbulence grids.

Plot2 generates streamline and vorticity contours as well as numerical output of the contours and grid points crossed.

Plot3 is the loads plotting program. Coefficient of pressure versus non-dimensionalized chord length for each selected angle of attack may be displayed, or plots for the

coefficients of lift, drag and moment versus angle of attack for unsteady cases are available.

IV. RESULTS AND DISCUSSION

A. COMPARISON WITH EXPERIMENTAL DATA

The first comparison was with experimental data from [Ref. 14] for a series of three different reduced frequencies, at the same Mach and Reynolds numbers. This was selected to highlight the time history dependent nature of dynamic stall. The plots pertaining to this comparison are enclosed in Appendix A. Considered were C_p versus non-dimensionalized chordlength or x/c, as well as C_L, C_m and C_D versus incidence angle, α.

Reduced frequency is a pitching rate parameter and is defined as:

$$rf = \frac{\omega C}{2C}$$

where:

ω = circular frequency,

C = airfoil chord length, and

U = freestream velocity.

The experimental data were taken at M = .072 to provide a valid approximation of incompressible flow.

For rf = .099, the plots of C_L, C_D and C_m versus α provide a good overview of the conditions the airfoil

experiences. In all three plots, the theoretical data follows the trends of the experimental data well, while the relative magnitudes and short term stall recovery features are less well represented.

The theoretical values of C_p also track relatively well with the experimental data, though slightly under-representing the pressures recorded experimentally. The theoretical results indicate the airfoil as beginning to stall at a slightly lower angle of attack than the experimental results do. At α = 20.8, the theoretical data show the vortex bubble as having been shed from the leading edge of the airfoil. Its progression across the upper surface can be followed in subsequent plots.

Streamline and vorticity contour plots are presented to illustrate the physical phenomena but are not directly compared with experimental data. Vorticity bubble generation and propagation are observed in each of the three cases.

For the plots of rf = .15, stall onset is noticably delayed from the rf = .099 case. Stall now occurs at $\alpha \approx 24^{o}$ for both the experimental and theoretical cases. As in the rf = .099 case, the trends show good correlation with post stall effects being less well modeled.

For rf = .248, the plots of C_L, C_D and C_m versus α have somewhat poorer correlation than the previous two cases.

There is greater offset between data, but stall occurs at $\alpha \approx 25^{\circ}$ for both data sets on the C_L versus curves.

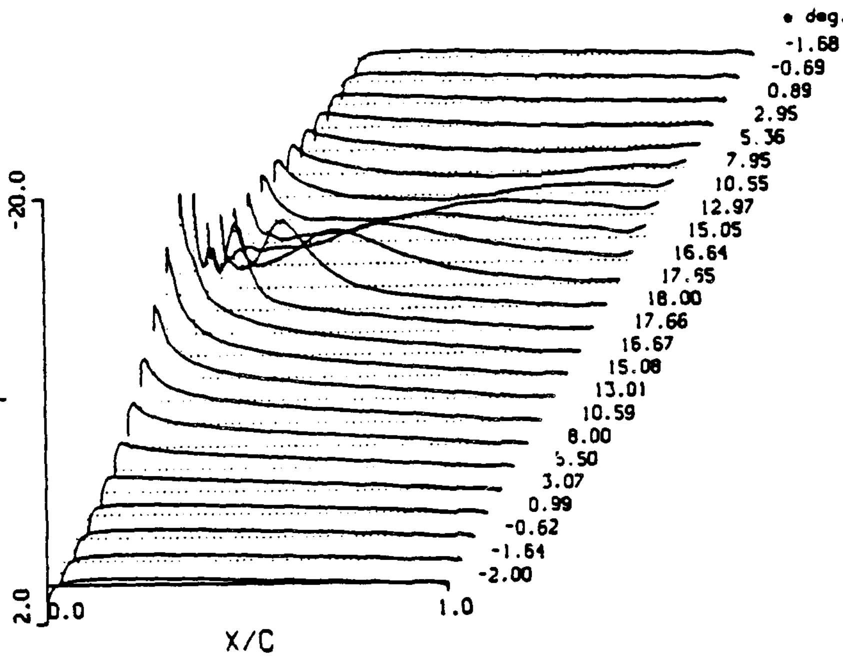
For these three cases, there is a slight trend towards higher maximum values on the C_L versus α curves with higher reduced frequency. Significantly, the theoretical results accurately reflect the trend towards a higher stall incidence angle with higher reduced frequency.

B. COMPARISON WITH OTHER THEORETICAL DATA

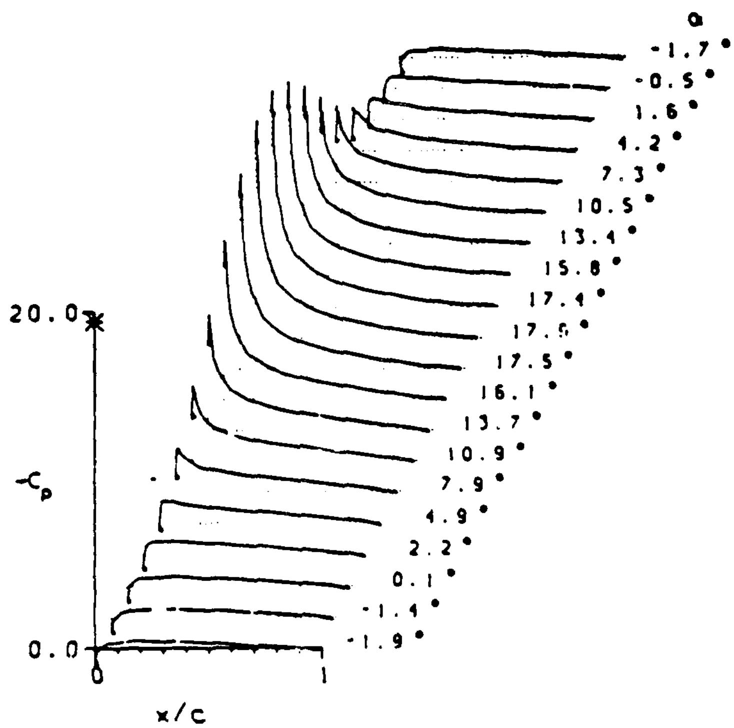
For the last case, a comparison was made between experimental data and two types of theoretical data. Experimental data are from [Ref. 14] and one version of the theoretical data is from [Ref. 15]. The conditions for this case were: reduced frequency = .199, mach number = .184 and $-2 \leq \alpha \leq 18^{\circ}$.

Figure 7 is the experimental and other theoretical data. Figure 8 is the theoretical data from the Wu code. Referencing the C_p versus x/c plots in Figs. 7a,b and 8c, the theoretical data from the Wu code demonstrate a very high degree of accuracy with the experimental data, while the theoretical data from the other source show a decidedly less accurate picture. The other source indicates dynamic stall and post stall conditions when experimentally, the airfoil never stalled.

Similar and pronounced differences also occur in the C_L and C_m versus α plots. As in the previous series, the Wu code data slightly under-represents the amplitude of the



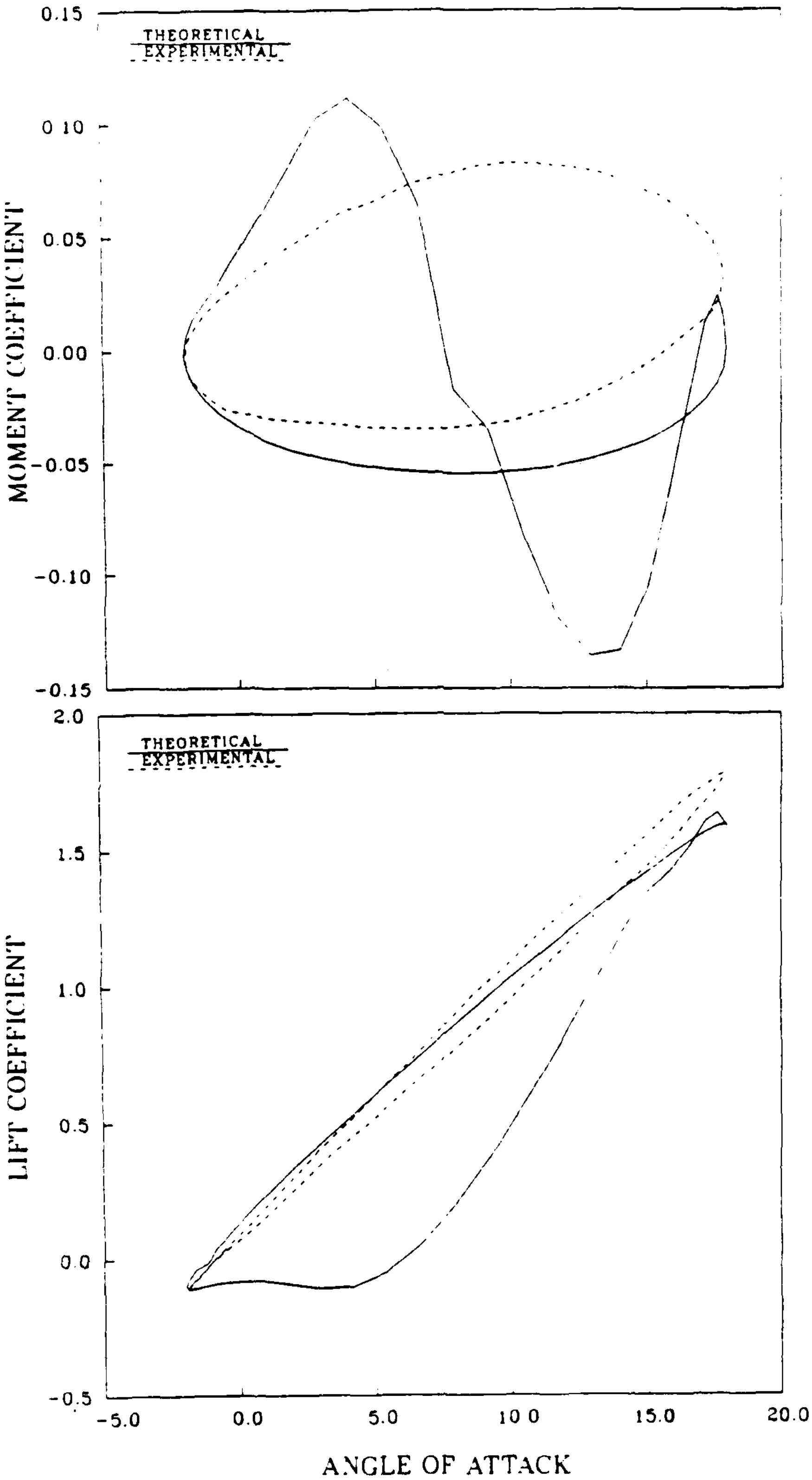
a.



b.

Figure 7a,b. C_p versus x/c for rf = .199, M = .184, Re = 2.45×10^6. Top = theoretical, Bottom = experimental.

c.



d.

Figure 7c,d. C_m and C_L versus α for rf = .199, M = .184, Re = 2.45×10^6

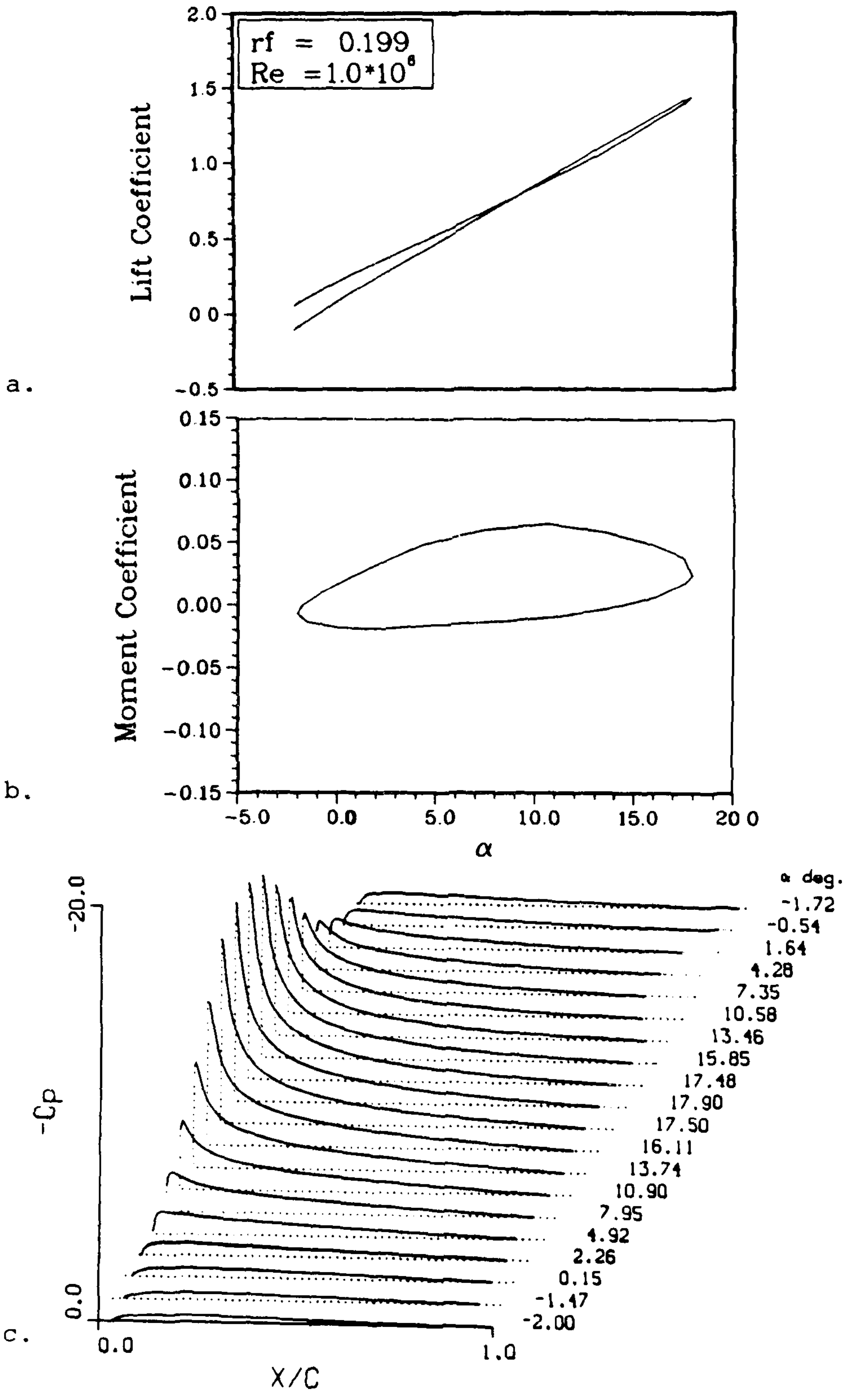


Figure 8. C_L, C_m versus α and C_p versus x/c for rf = .199, M = 0, Re = 2.45 × 10^6

pressure but still follows the trends well, while the other theoretical data which incorrectly indicates a dynamic stall condition.

For this case, the data from the Wu code data are a more accurate reflection of the experimental data than the data from the other theoretical source.

C. ENHANCEMENTS TO THE CODE

There were two primary enhancements that were added to the Wu code. These are:

1. The ability to output plotting data at non-regular intervals, to add flexibility to the code and to simplify and facilitate comparison with experimental data.

2. The availability of the code to generate physical plane velocity data to allow the use of Plot3D and its associated graphics functions.

The modifications to provide physical plane velocity information centered around the transformation matrix that is used in GEOM to obtain the physical plane grid from the computational plane grid after the computational grid is generated.

The scale factor of the transformation is defined as

$$H = \left|\frac{\partial Z}{\partial \zeta}\right|$$

where Z is the physical plane and

$$Z = re^{i\theta}$$

$$= f(\zeta)$$

$$Z = f(\rho e^{i\phi})$$

while ζ is the computational plane, where the airfoil is represented by the unit circle and cylindrical coordinates are employed.

Then, denoting $\frac{\partial Z}{\partial \zeta}$ as DZDW, HSTAR is defined as:

H* = complex(real(DZDW),-absolute imaginary (DZDW))

HSTAR is written onto tape 20 and passed to ZONST, where it is used in the subroutine KMTCS. In KMTCS, two additional arrays are defined, VTOTCO and VTOTPH, which are the total computational plane and physical plane velocities, respectively. They are defined as:

VTOTCO = complex (W1,W2) and

VTOTPH = VTOTCO/HSTAR

where W1 and W2 are the computational plane velocities in the rotating frame of reference. Additional minor modifications to the code were made to implement these changes.

Representative plots are presented for four cases where the reduced frequency is .150 and Reynolds number is 1×10^6.

1. Laminar flow with the airfoil at a steady state 5.0° α. This is portrayed in Fig. 9.

a.

b.

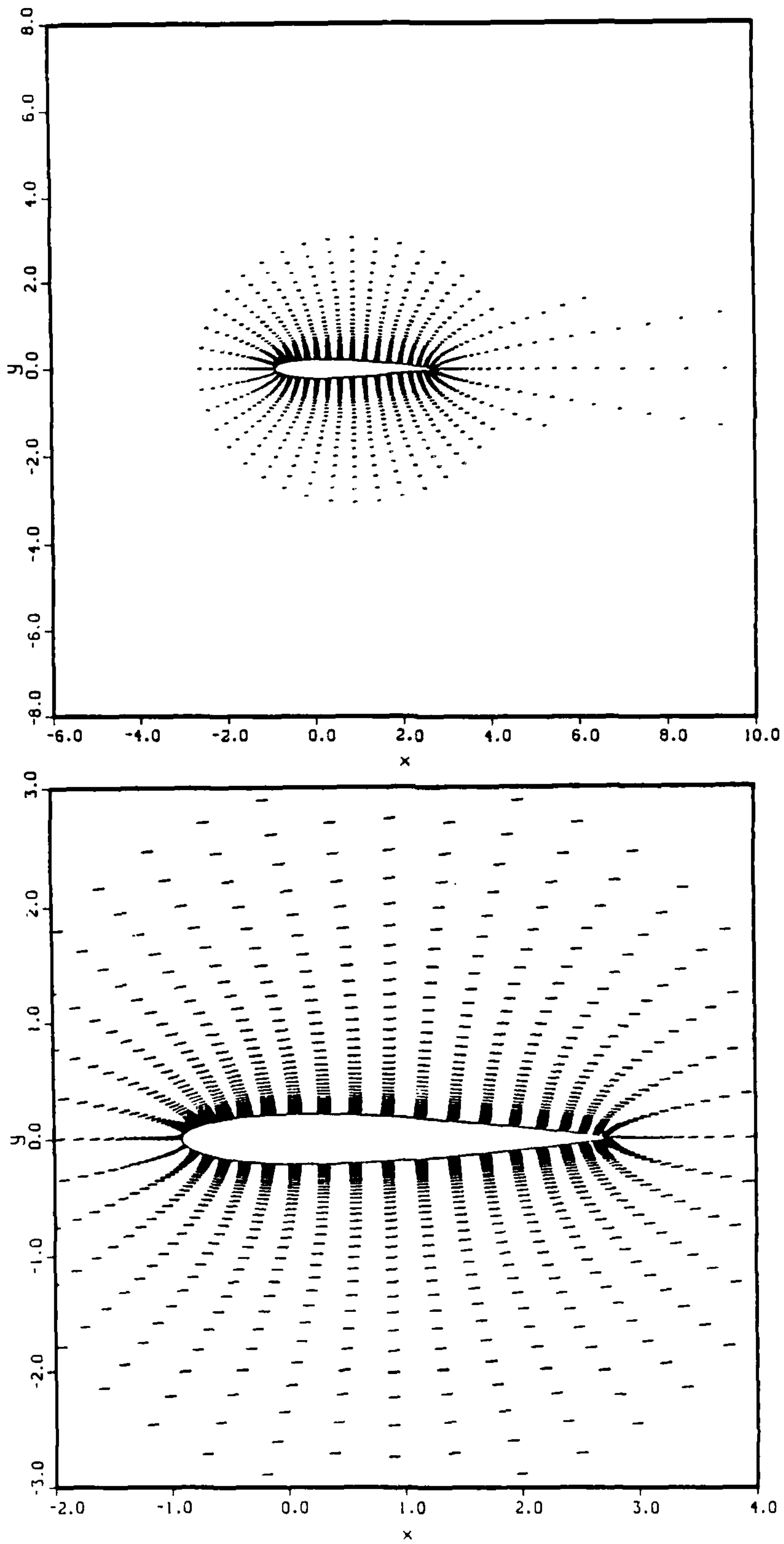
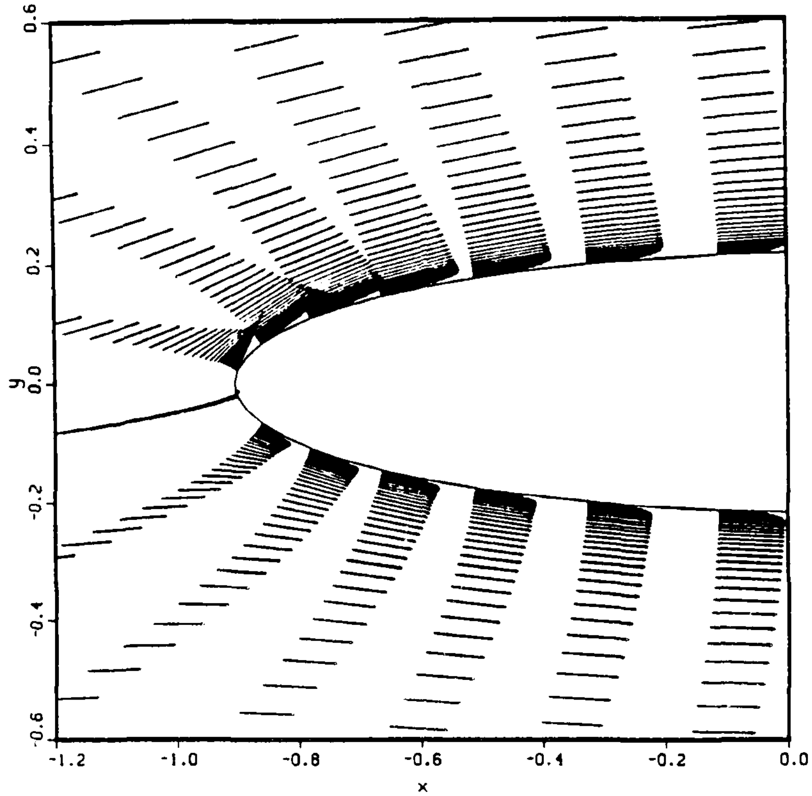


Figure 9. Velocity Vector Function for 5° α

c.



d.

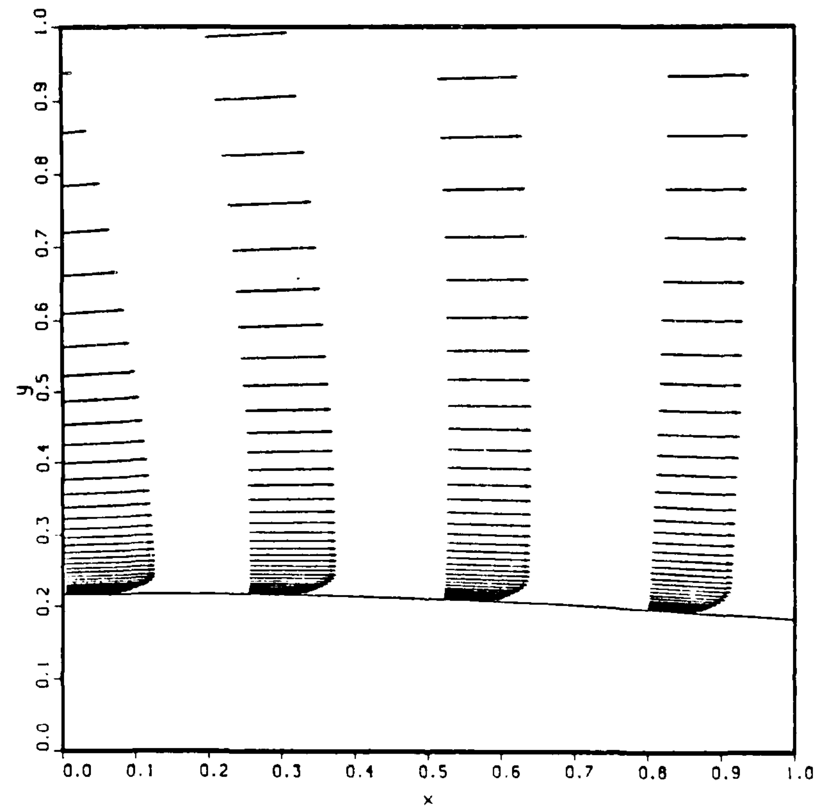
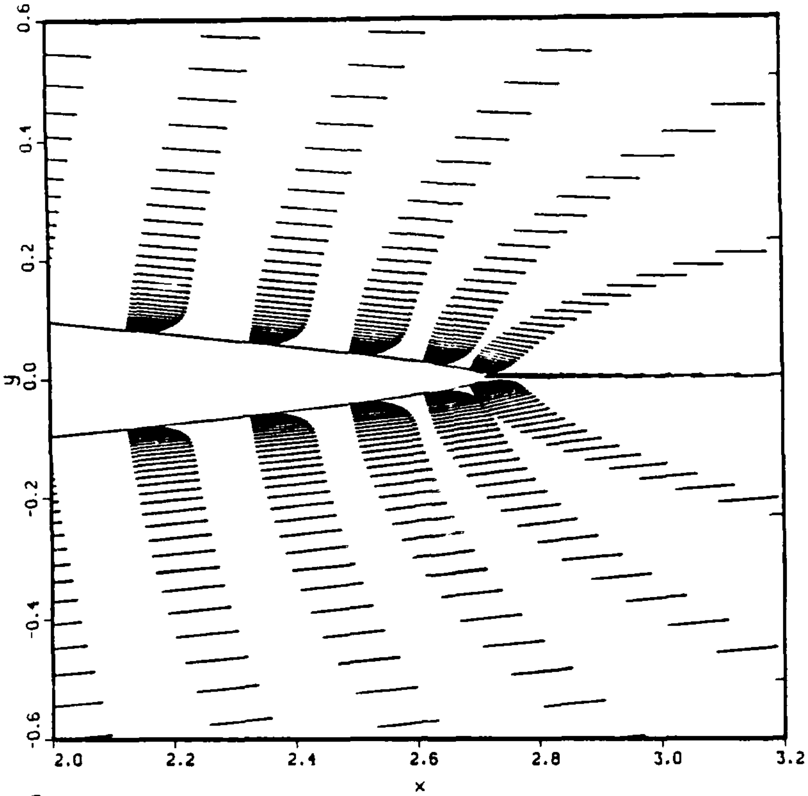


Figure 9 (Continued)

e.



f.

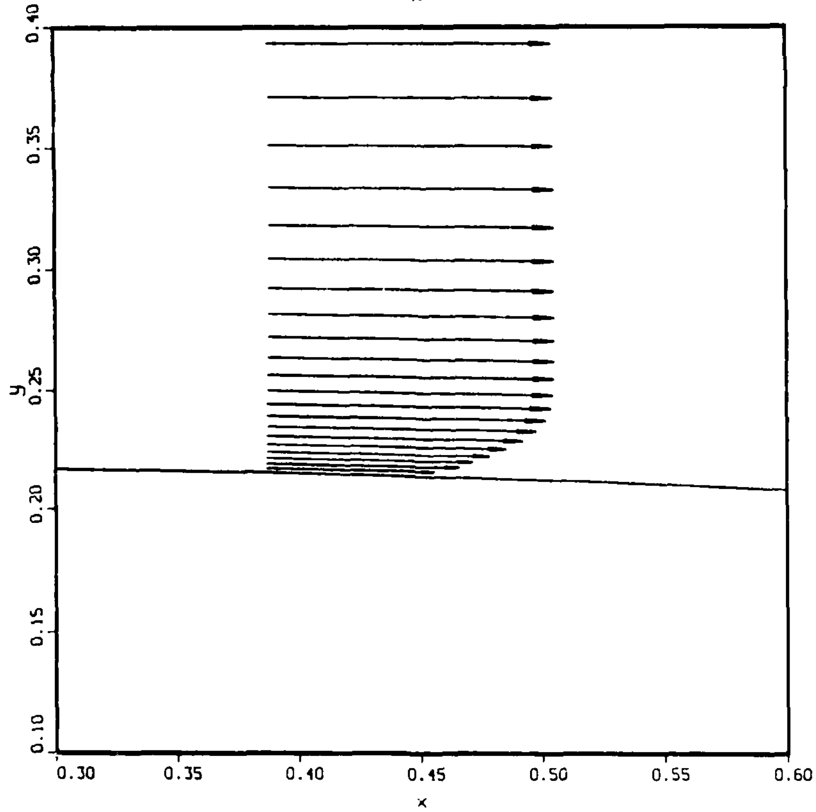


Figure 9 (Continued)

2. Flow conditions immediately prior to stall onset are shown in Fig. 10.

3. Initial indication of reverse flow at 21.06° α in Fig. 11. This appears on the upper surface near the leading edge as the vorticity bubble starts to form, which quickly spreads over the entire upper surface of the airfoil.

4. Turbulent flow with the airfoil at 23.8° α, shortly after the onset of dynamic stall. This is portrayed in Fig. 12. In all cases, alternate radial grid lines were deleted from the plots to enhance clarity, while a similar display progression is followed for the first and last cases.

Two additional modifications would help interpretation of the information. First, computing VTOTPH for the entire physical plane, not just in the viscous region. Second, rotating the grid so that the airfoil is graphically shown at its true angle of incidence. Both of these deficiencies are presented in Fig. 9a, and neither affects the accuracy of the graphical output in the viscous region.

In Fig. 9a, the primary portion of the viscous region and part of the inviscid region are shown.

Subsequent portions of Fig. 9 show increasing resolution of various portions of the plot in Fig. 9a. To aid in comparison, the horizontal scale is consistent throughout the velocity plots.

Fig. 9c shows the leading edge stagnation point and tangency of the boundary layer at the surface of the airfoil. Fig. 9d is the midchord upper surface region. Fig. 9e is the aft portion of the airfoil. Fig. 9f is a

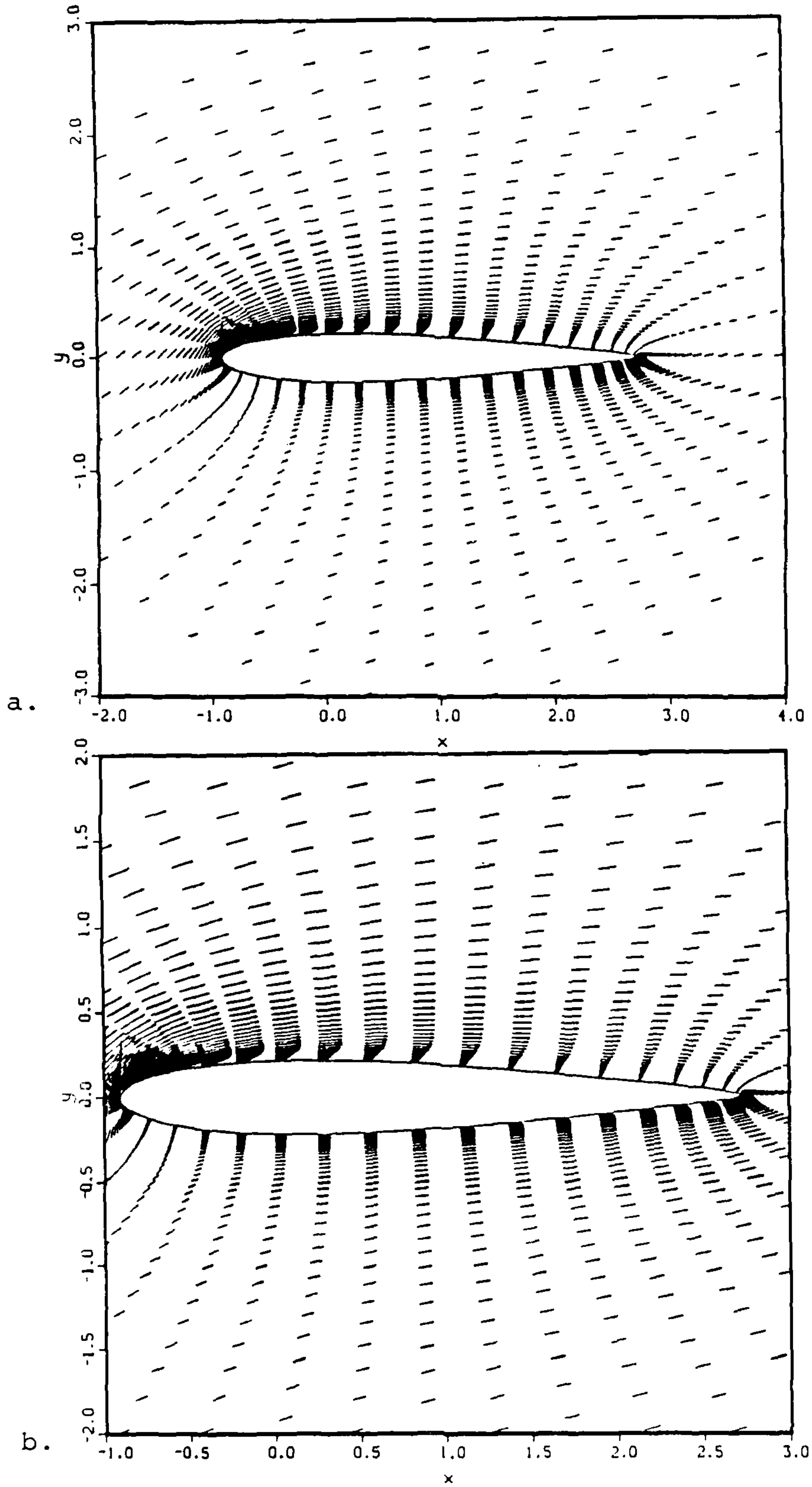


Figure 10. Velocity Vector Function for α = 20.24°

a.

b.

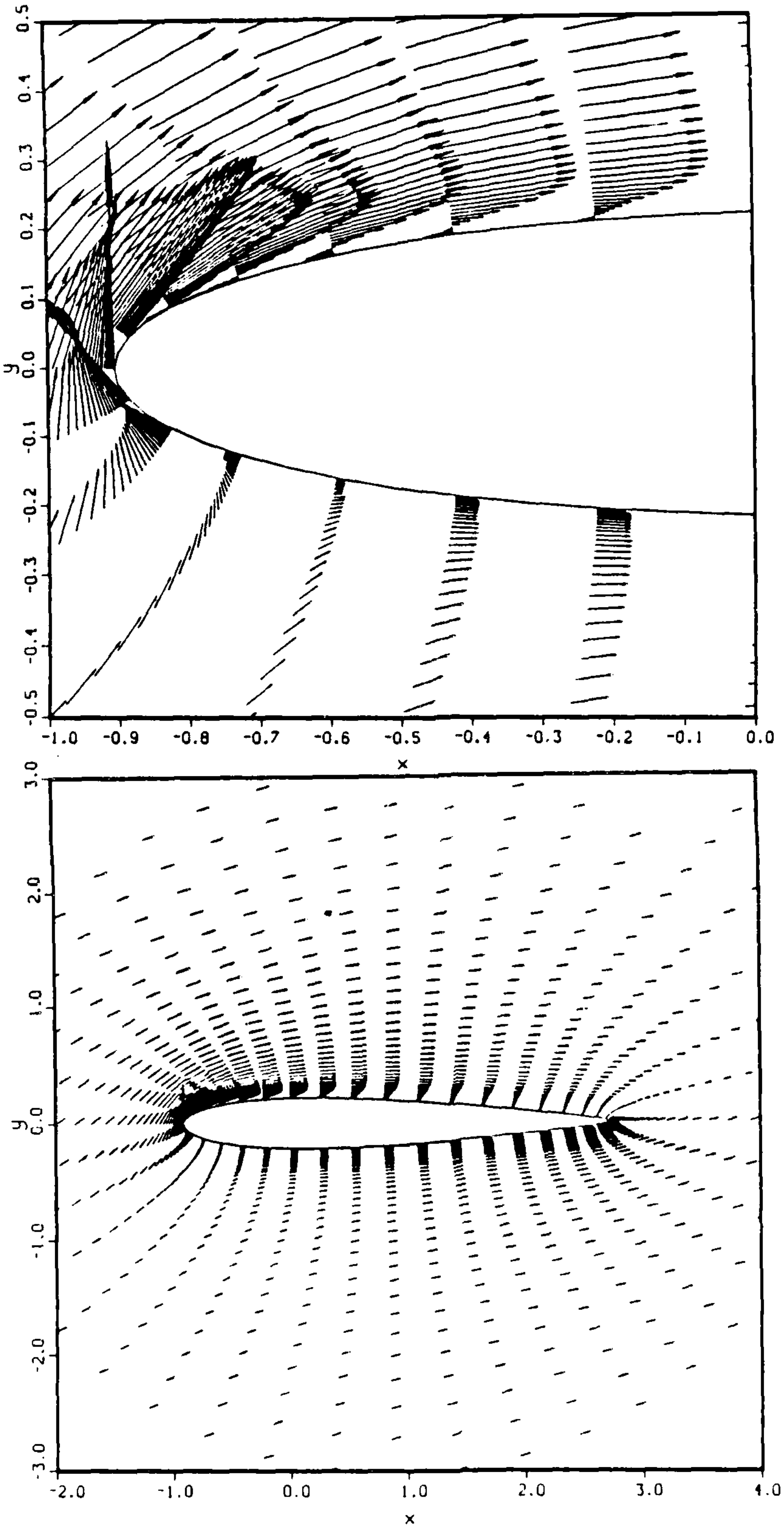


Figure 11. Velocity Vector Function for α = 21.06°

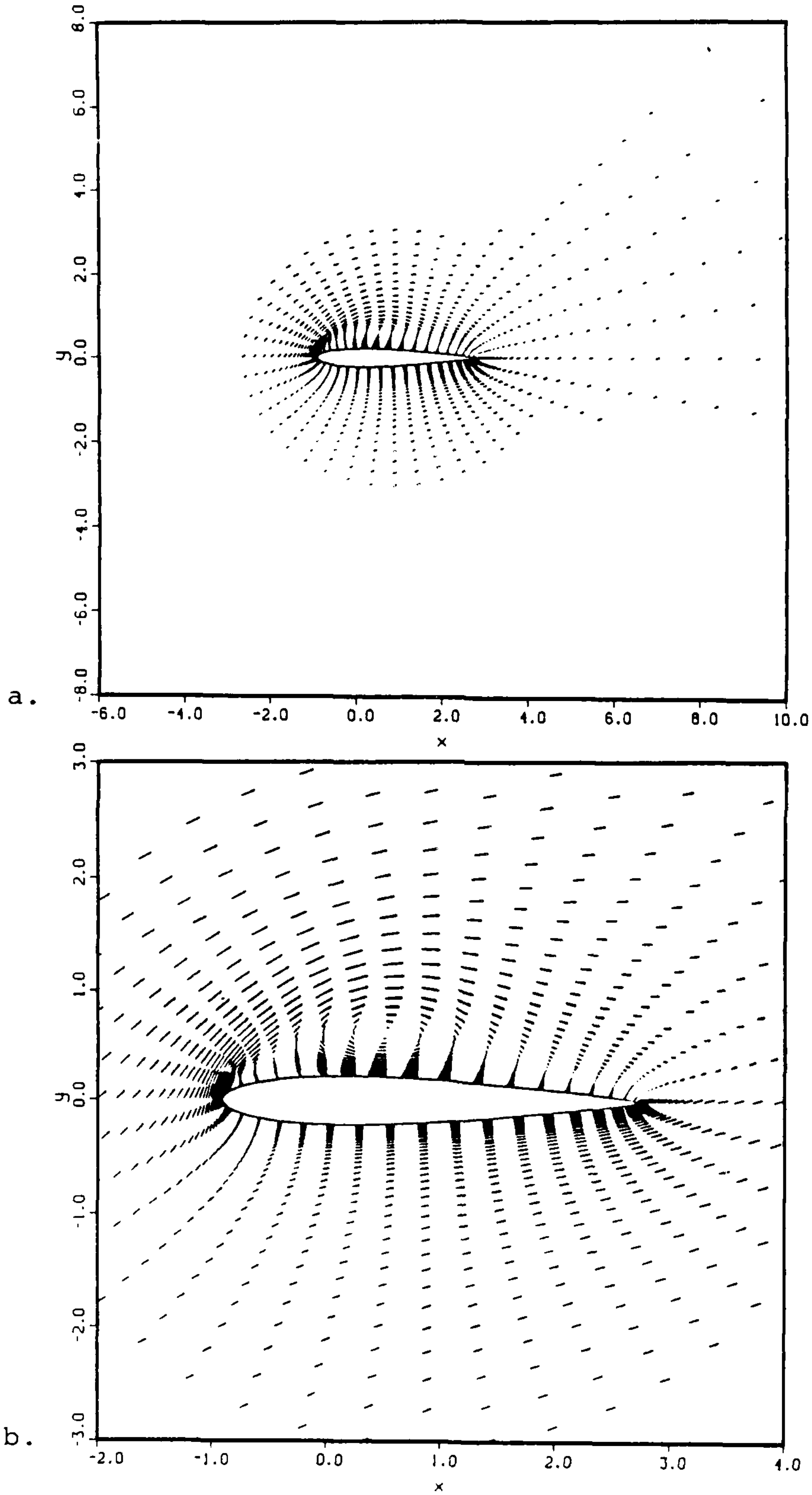
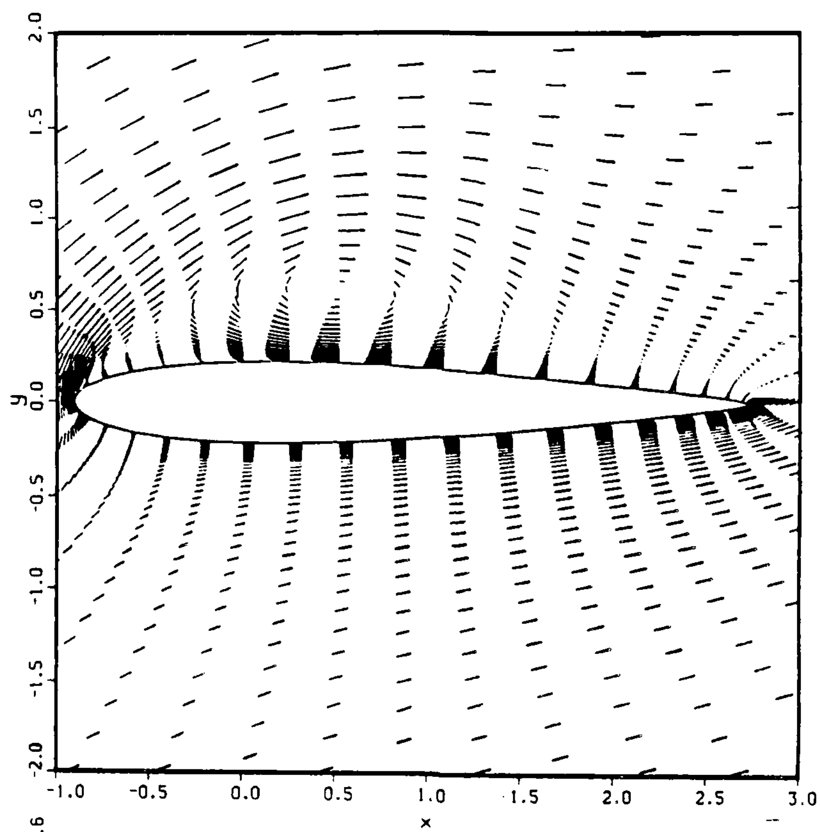


Figure 12. Velocity Vector Function for α = 23.83°

c.



d.

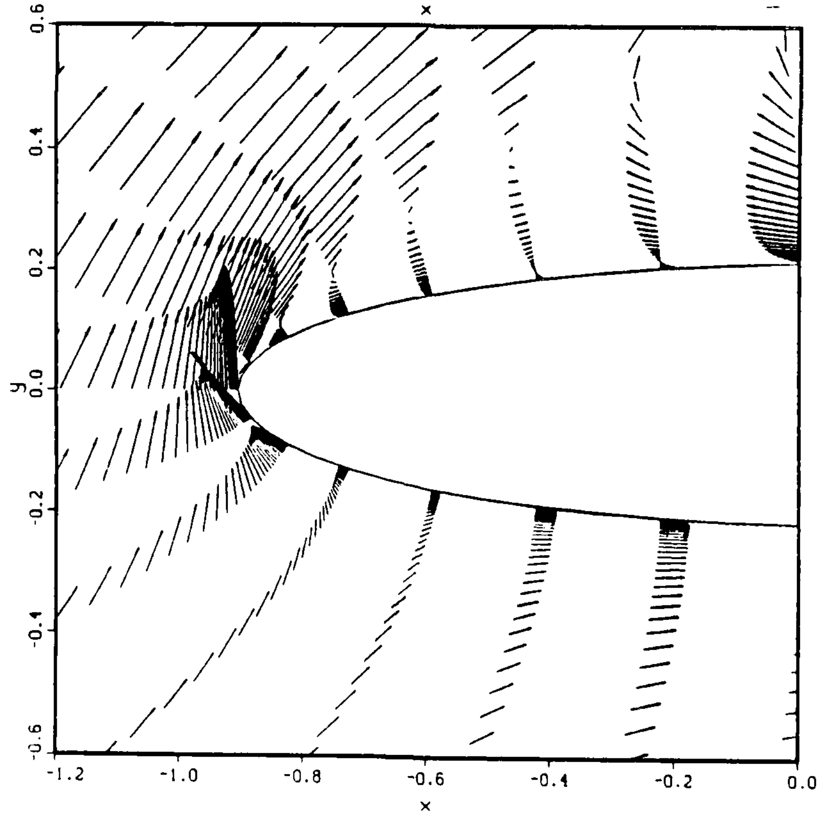
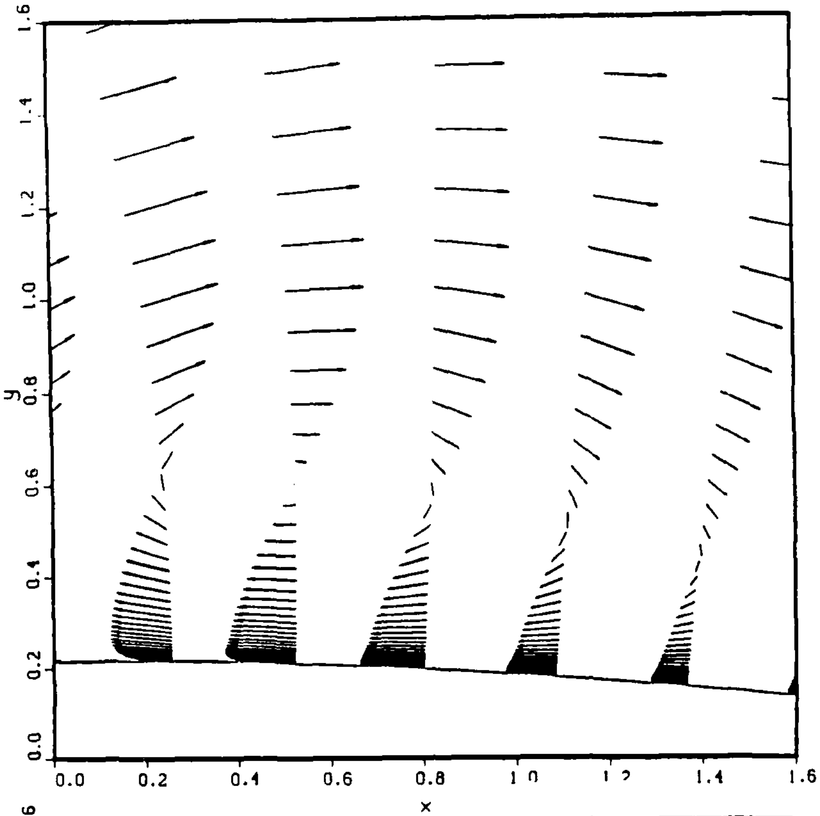


Figure 12 (Continued)

e.



f.

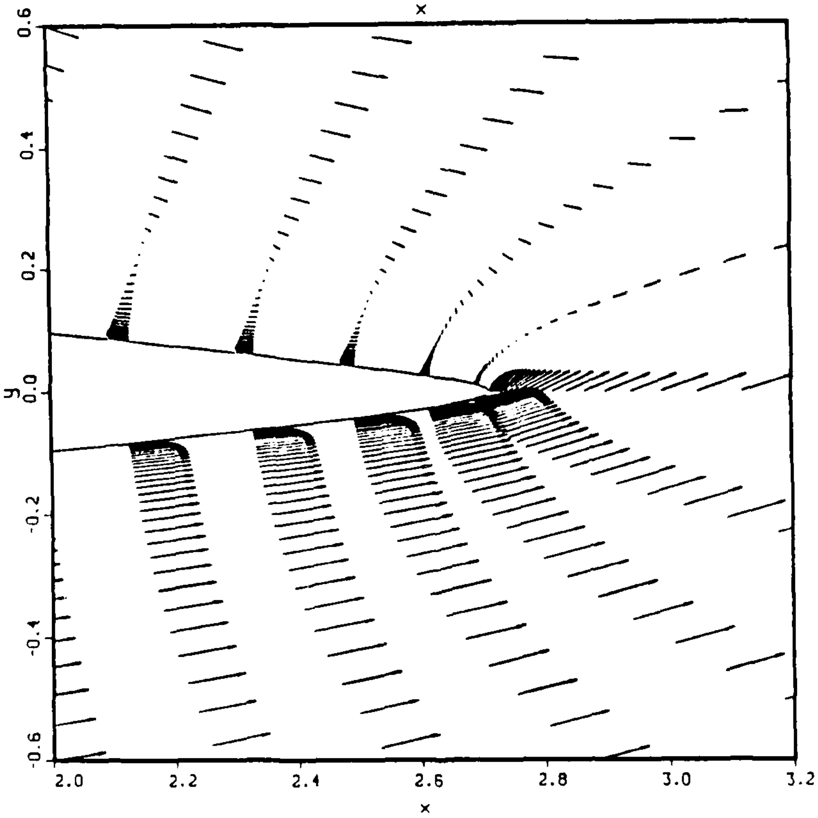


Figure 12 (Continued)

detail plot of the boundary layer near the mid-span, upper surface.

Fig. 11 is the case for dynamic stall. Flow reversal is apparent over the entire upper surface, with the vortex bubble center indicated by the zero velocity vector located above approximately .3 chord. In Fig. 11c, the high velocity gradients are readily apparent, from the leading edge around to the upper surface.

More complex flow patterns can be readily portrayed.

V. CONCLUDING REMARKS

The Wu code holds much promise to help unlock some of the mysteries of dynamic stall. Its strengths include the following:

- Enough speed and efficiency that it can be operated on a VAX or similarly-sized computer.
- Good success at indicating the trends of the cases studied.
- Relatively accurate results.
- Powerful diagnostic tool which can become a predictive tool.
- Not Reynolds number limited.

Other aspects that should be noted are:

- The mathematical formulation is more involved than a straight finite differencing of pressure/velocity Navier-Stokes equations.
- As currently formulated, it is not readily applicable to arbitrary geometries, but a useful selection of Joukowski transforms is available.

With the addition of compressibility and transition modeling and later, extension to three dimensional representation, its utility will continue to be enhanced and its applications expand.

APPENDIX A

REDUCED FREQUENCY VARIATION PLOTS

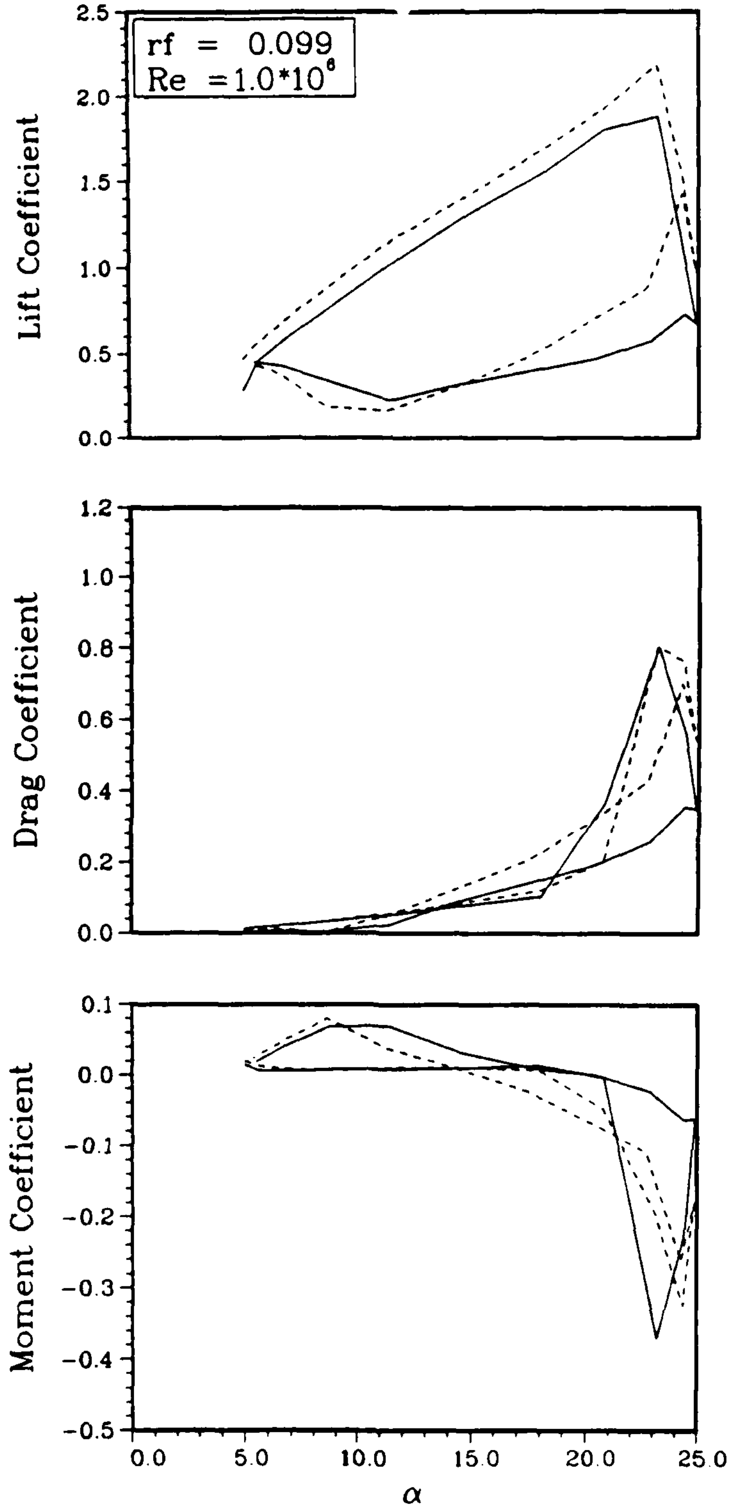
Appendix A contains plots for variation of reduced frequency. For each of the three reduced frequencies, the order is: C_L, C_D and C_m versus α

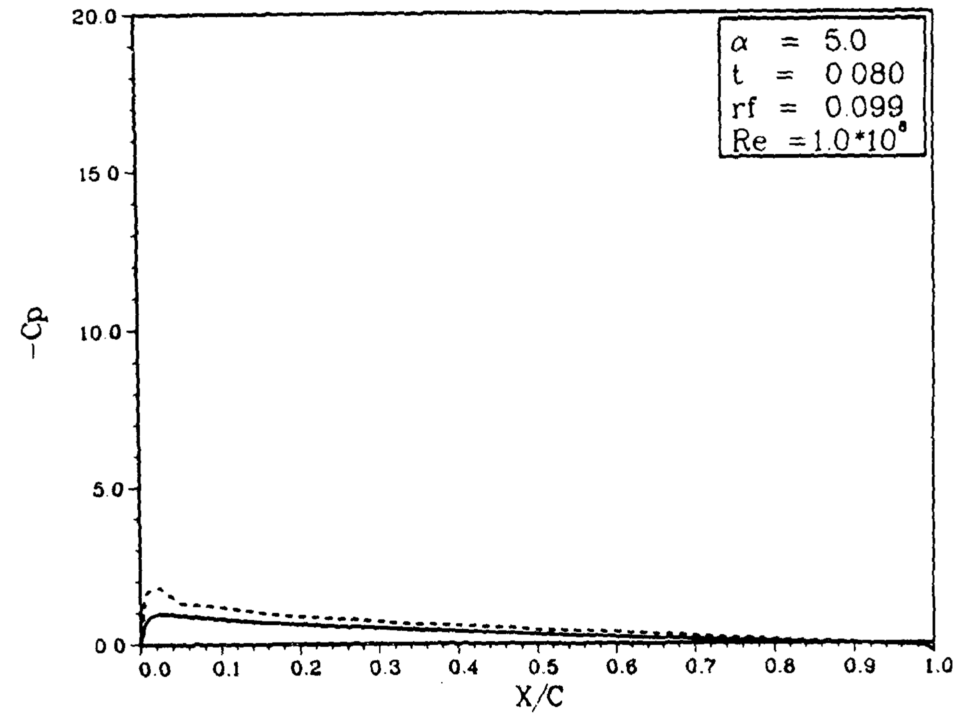
C_p versus x/c with varying α

Streamline and vorticity contours with varying α.

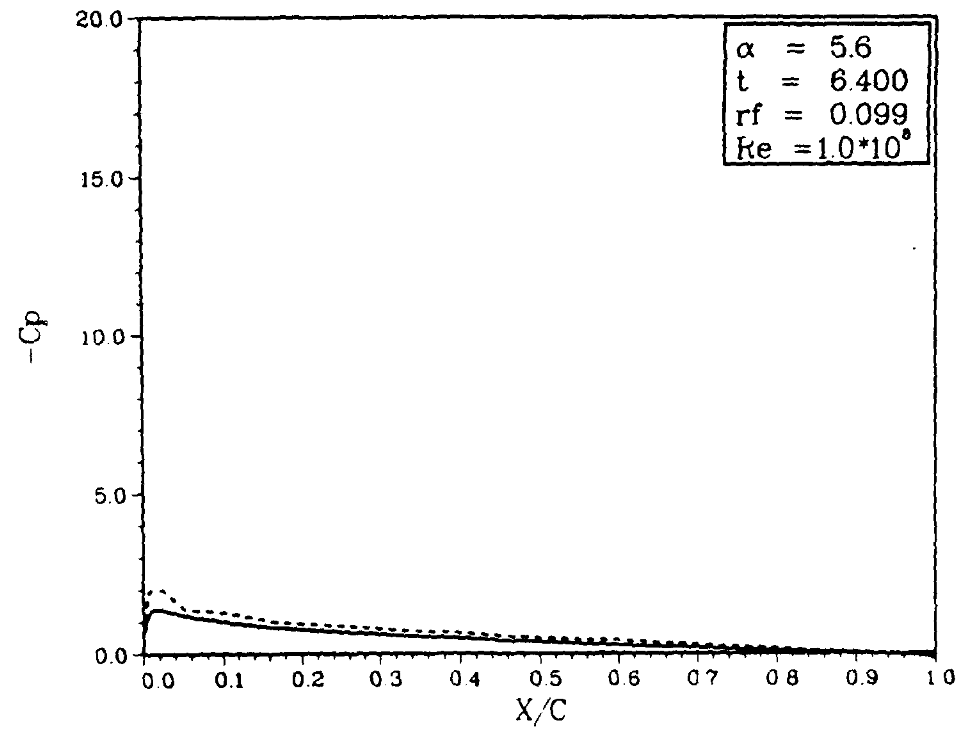
Theoretical: ______________

Experimental: -------------

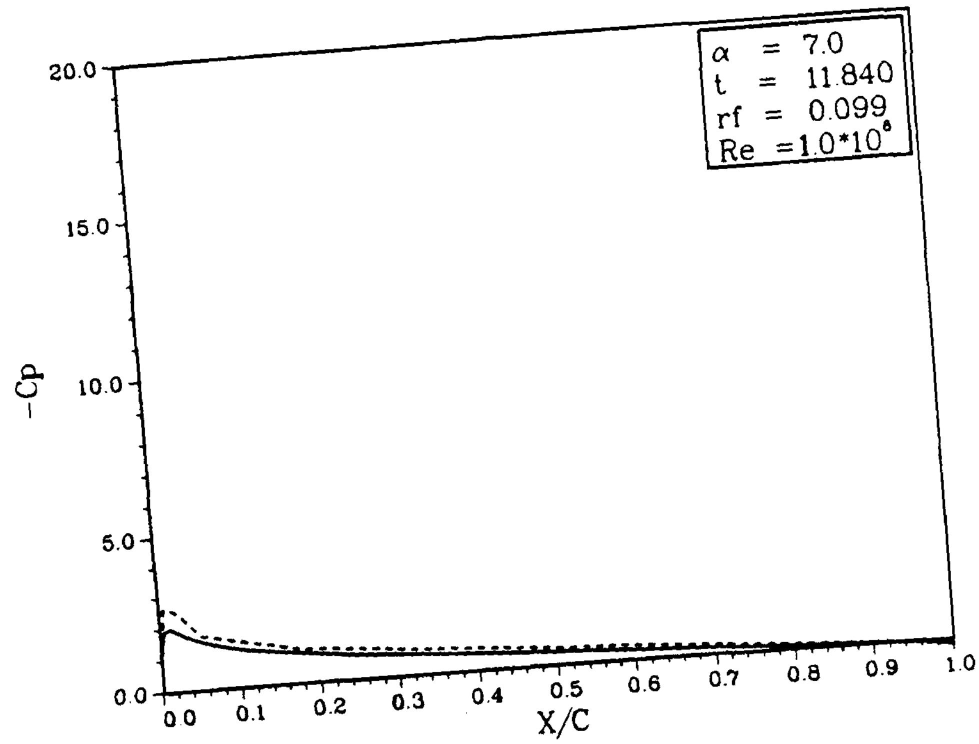

rf = 0.099
Re =1.0*10^6
Lift Coefficient
Drag Coefficient
Moment Coefficient
α
0.0
5.0
10.0
15.0
20.0
25.0

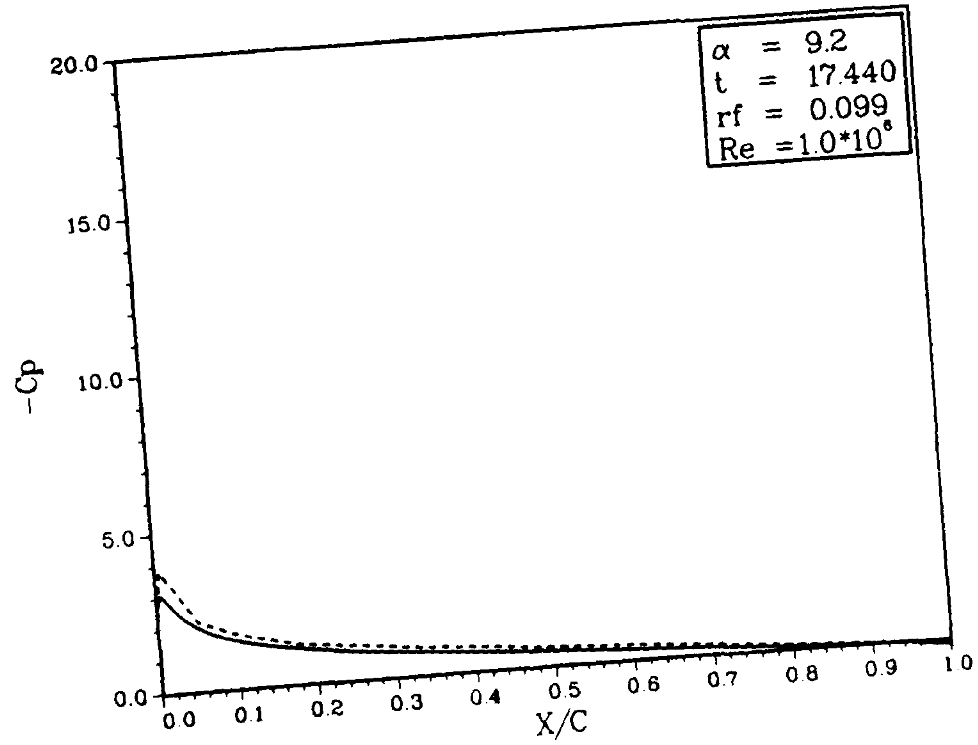


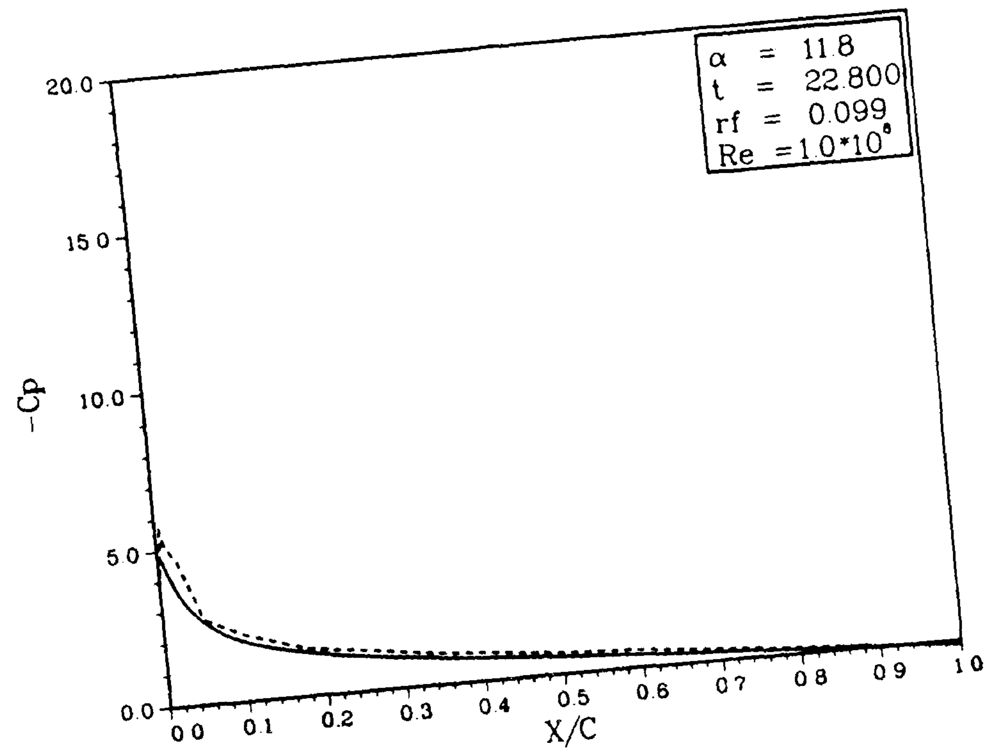
α = 5.0
t = 0.080
rf = 0.099
Re =1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

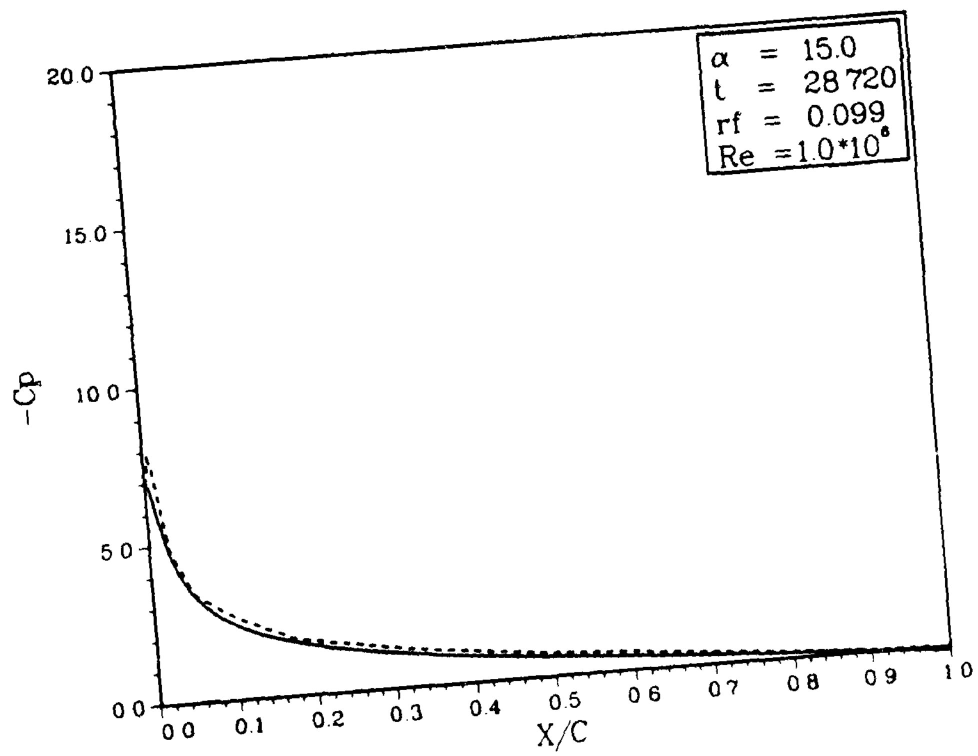


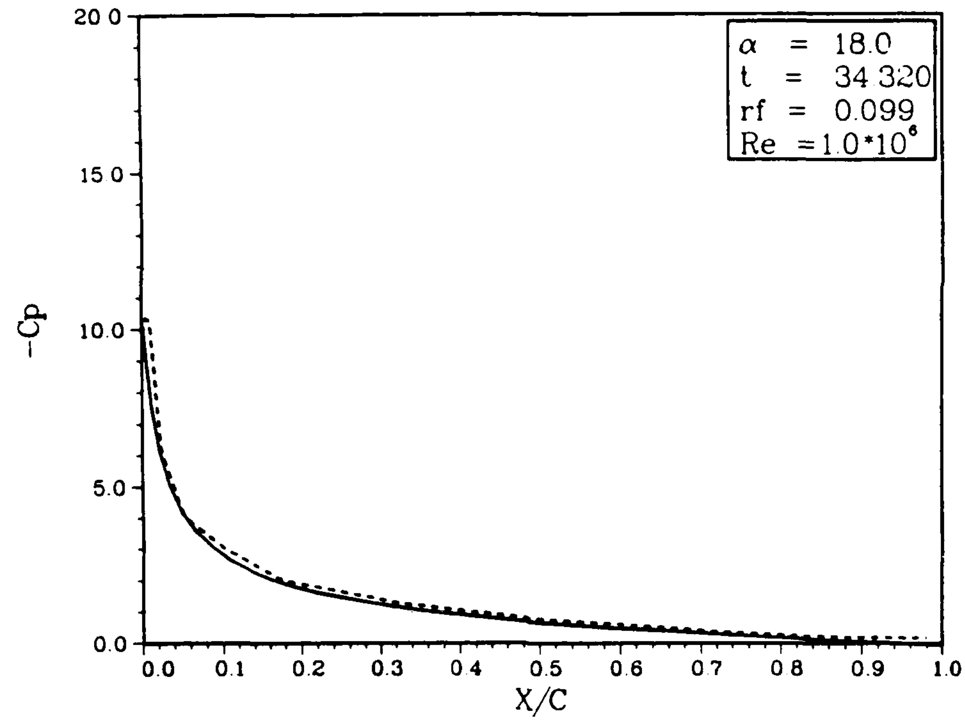
α = 5.6
t = 6.400
rf = 0.099
Re =1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C


α = 7.0
t = 11.840
rf = 0.099
Re =1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

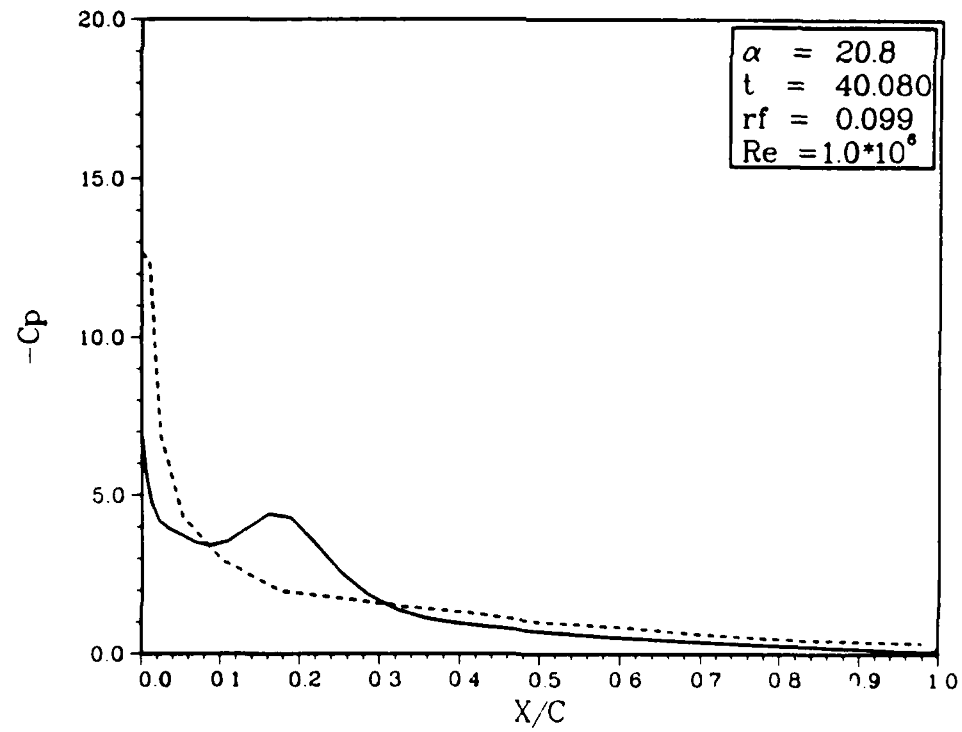

α = 9.2
t = 17.440
rf = 0.099
Re =1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


α = 11.8
t = 22.800
rf = 0.099
Re =1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C

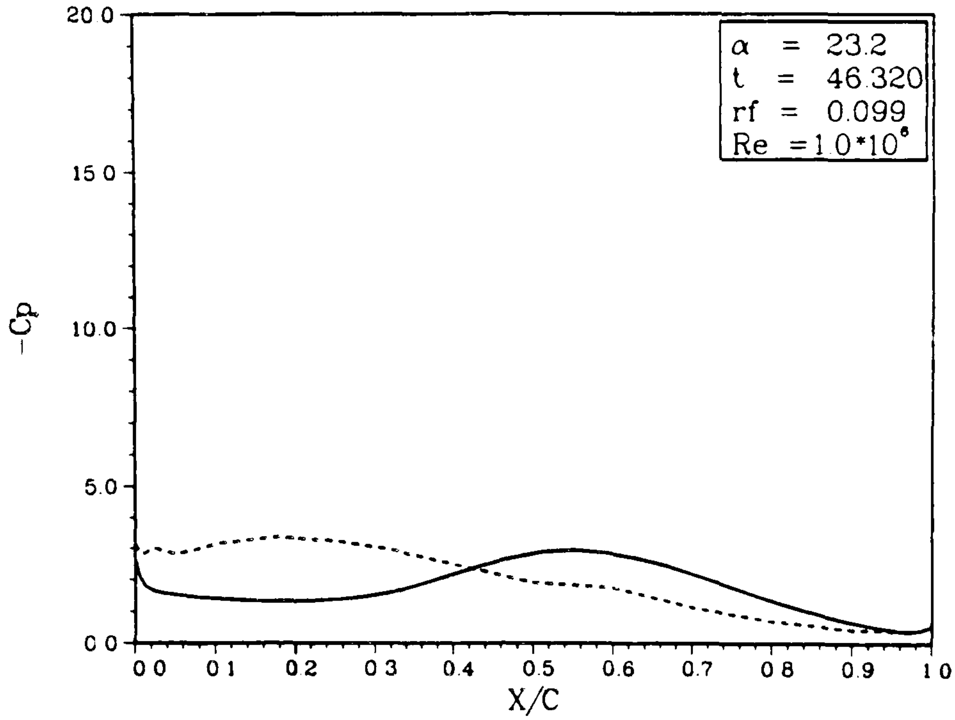

α = 15.0
t = 28.720
rf = 0.099
Re =1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C

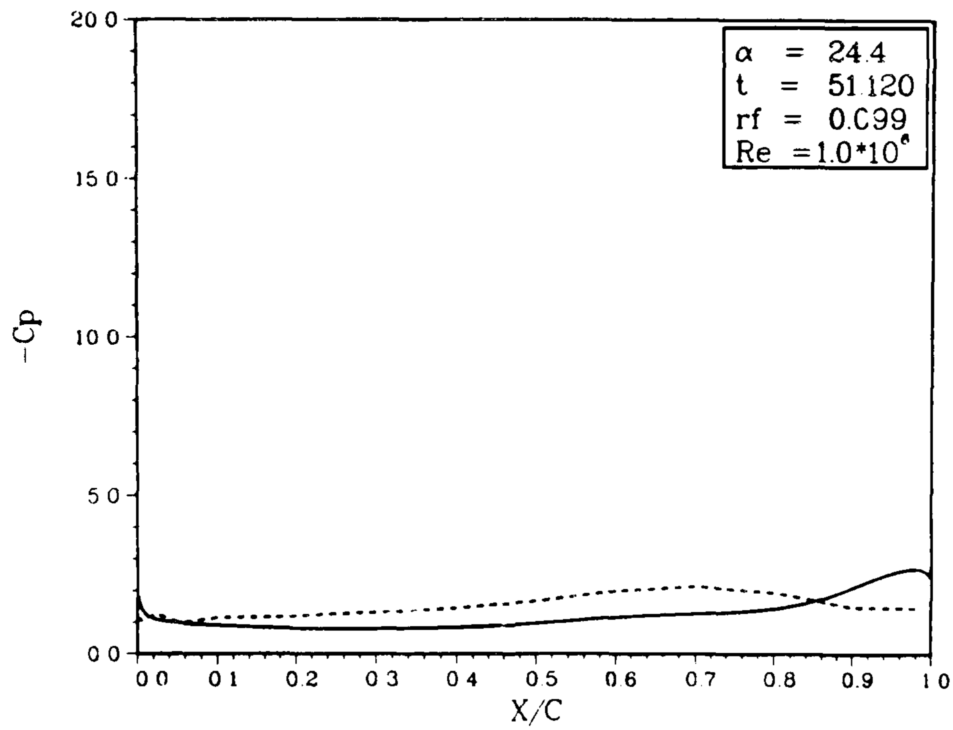


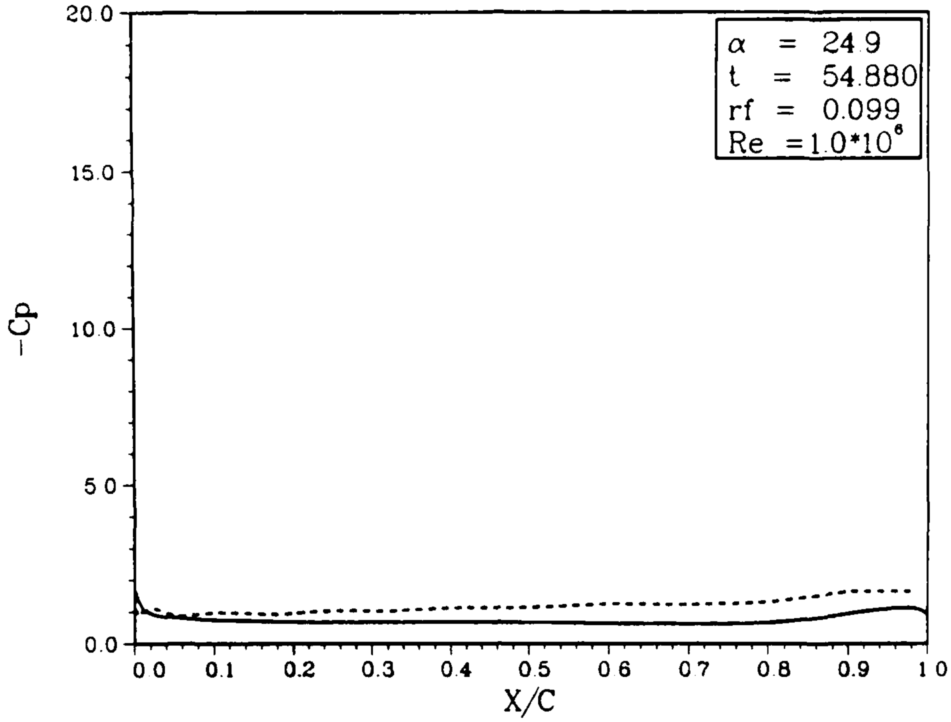
α = 18.0
t = 34.320
rf = 0.099
Re =1.0*10^6
-Cp
X/C



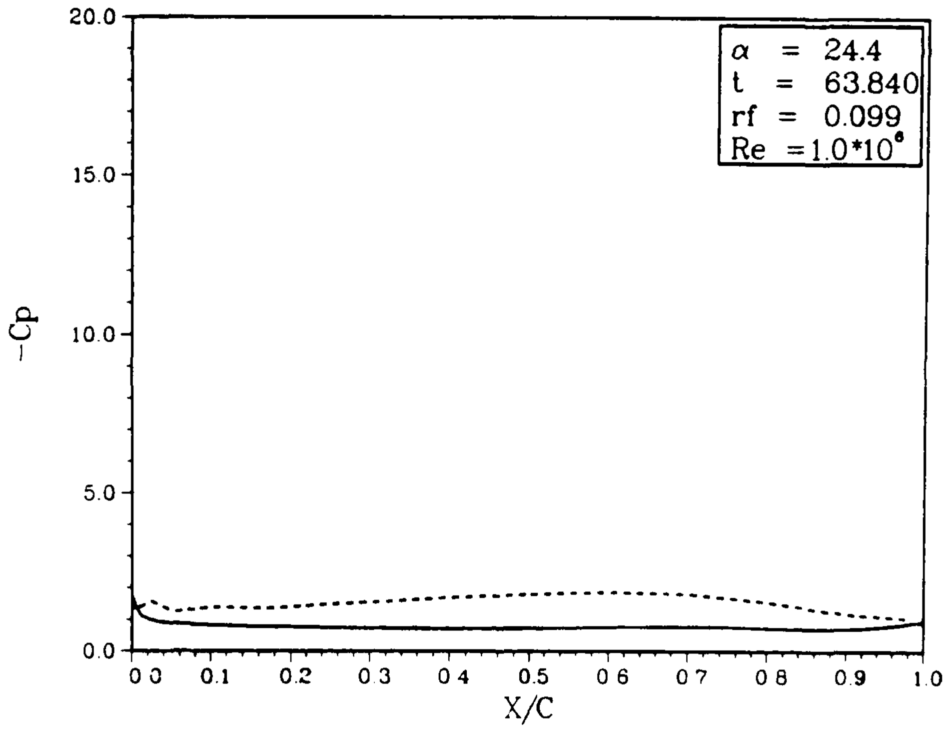
α = 20.8
t = 40.080
rf = 0.099
Re =1.0*10^6
-Cp
X/C


α = 23.2
t = 46.320
rf = 0.099
Re =1.0*10^6
-Cp
X/C
20 0
15 0
10.0
5.0
0 0
0 0
0 1
0.2
0 3
0.4
0.5
0.6
0.7
0 8
0.9
1 0

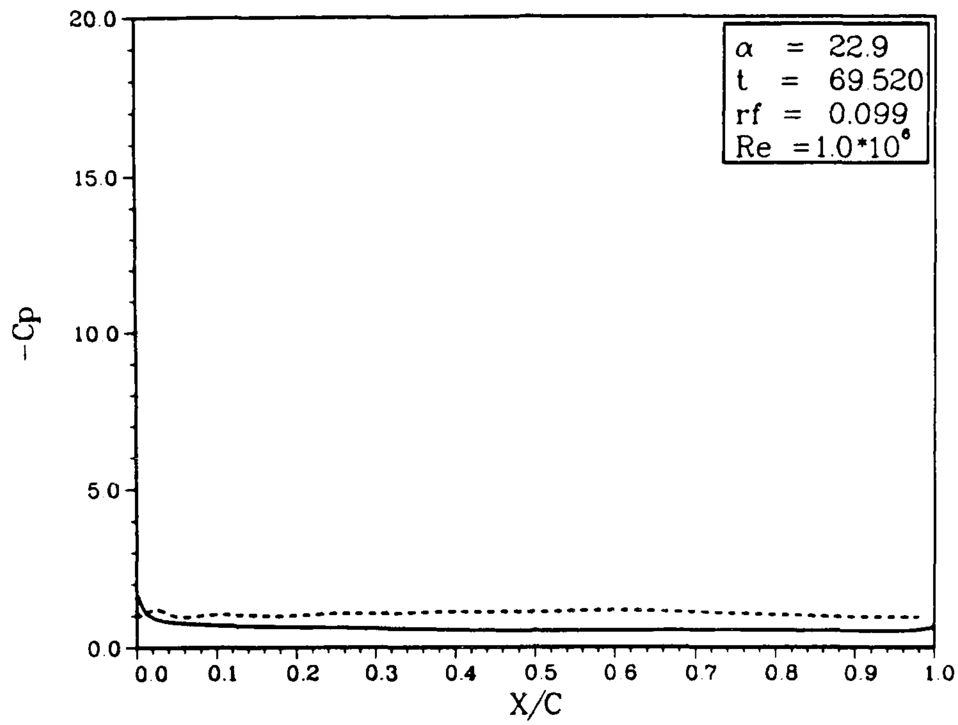

α = 24.4
t = 51.120
rf = 0.C99
Re =1.0*10^6
-Cp
X/C
20 0
15 0
10 0
5 0
0 0
0 0
0 1
0.2
0 3
0 4
0.5
0.6
0.7
0.8
0.9
1 0



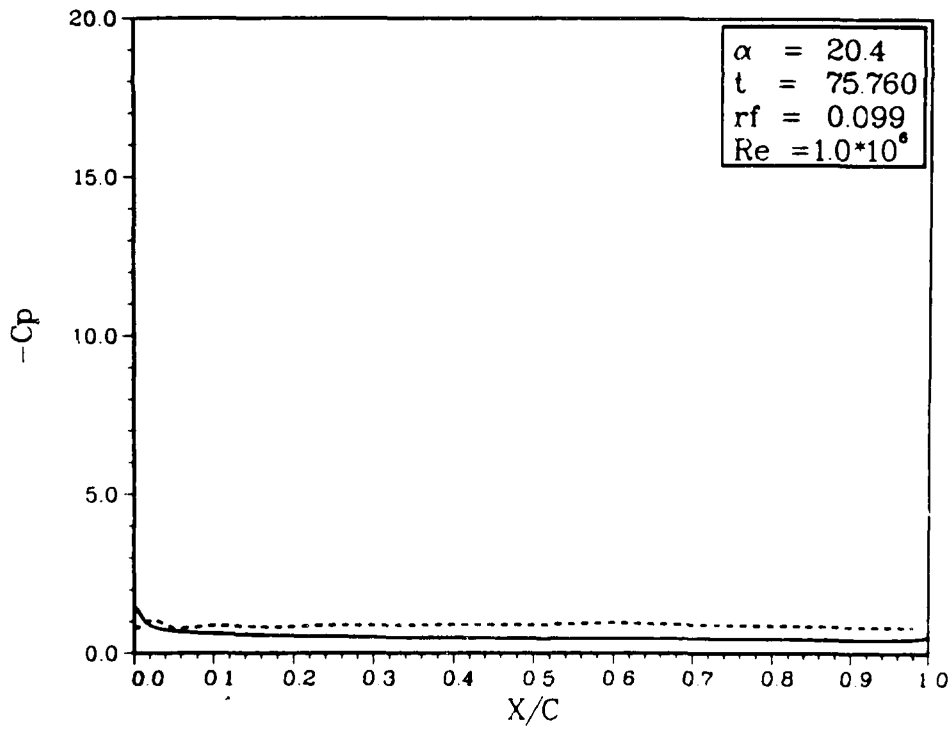
α = 24.9
t = 54.880
rf = 0.099
Re = 1.0*10^6
-Cp
X/C



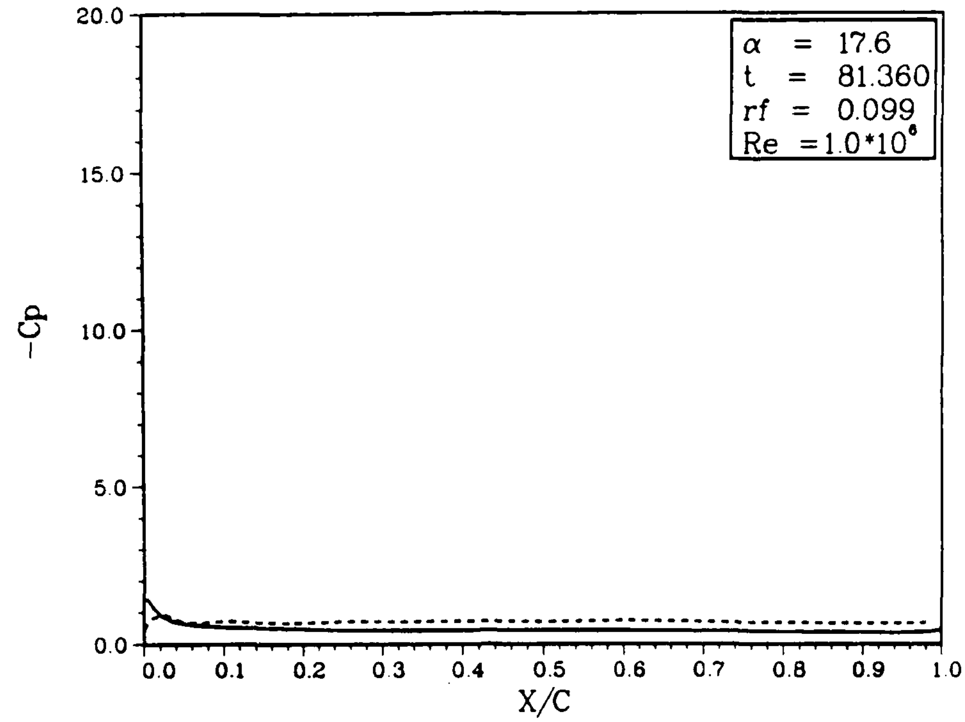
α = 24.4
t = 63.840
rf = 0.099
Re = 1.0*10^6
-Cp
X/C



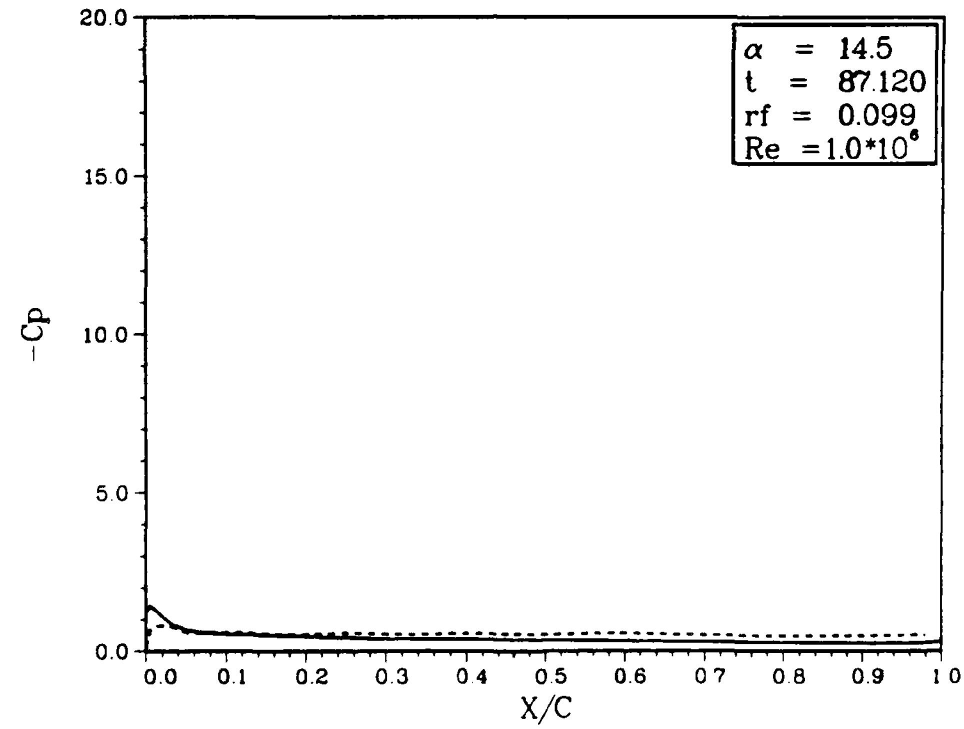
α = 22.9
t = 69.520
rf = 0.099
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C



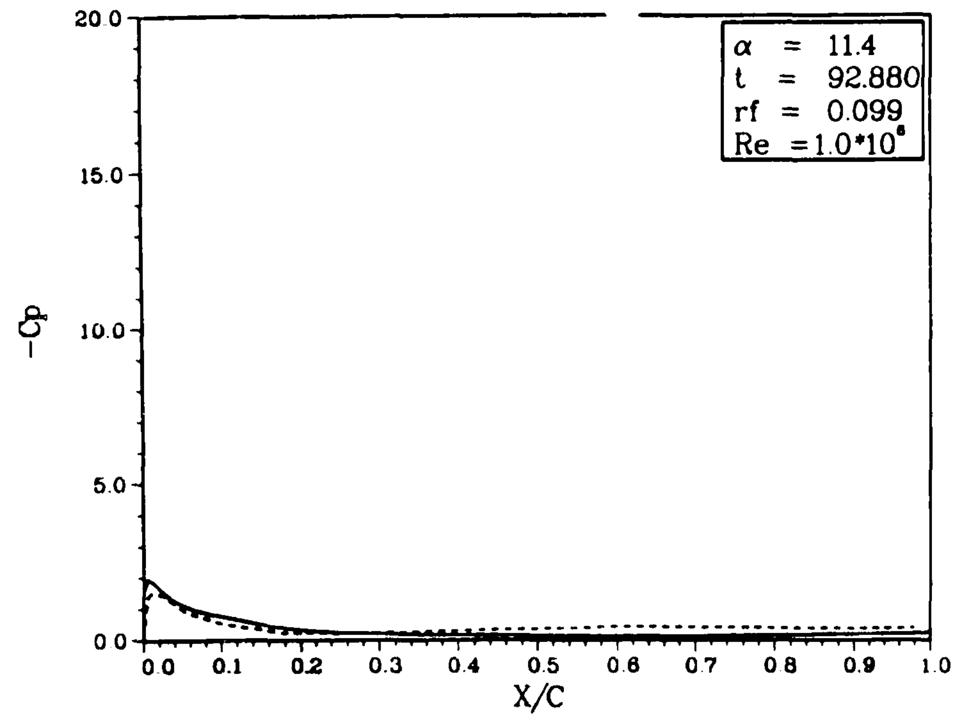
α = 20.4
t = 75.760
rf = 0.099
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C

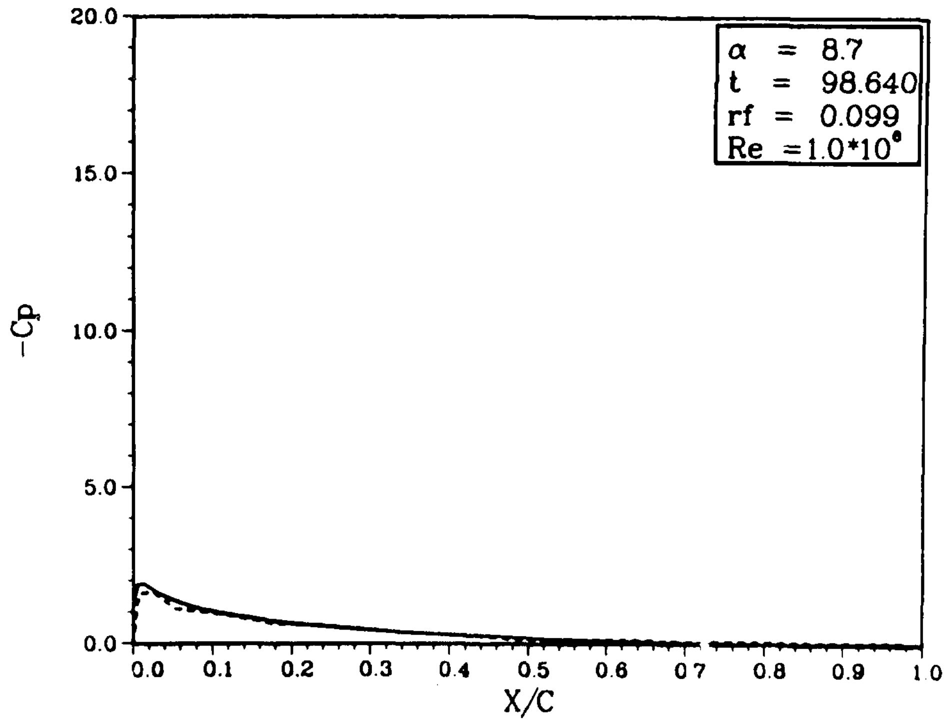


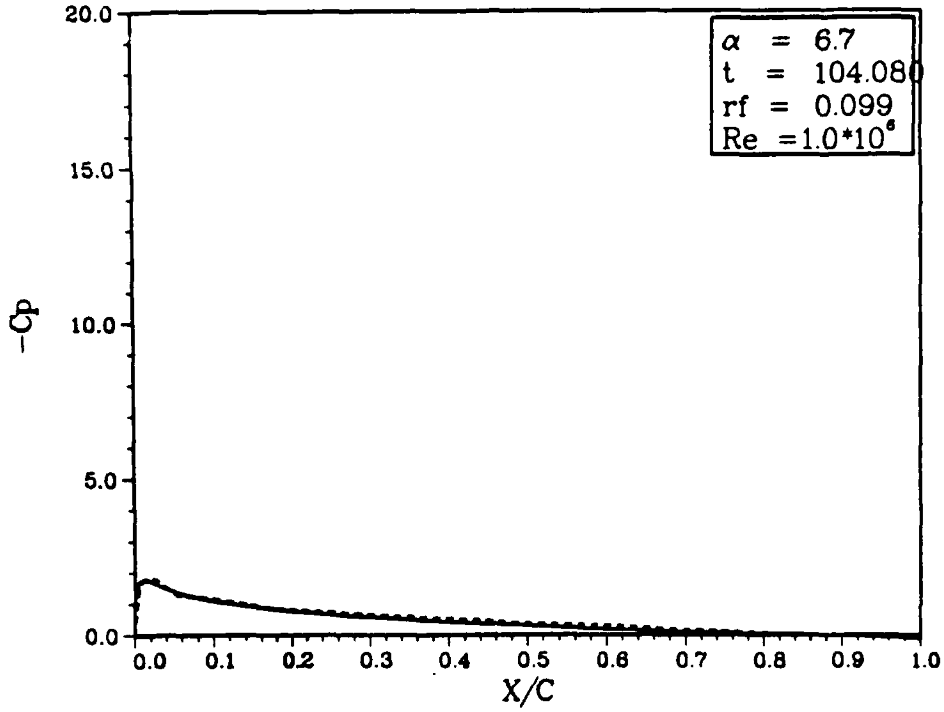
α = 17.6
t = 81.360
rf = 0.099
Re = 1.0*10⁶
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0



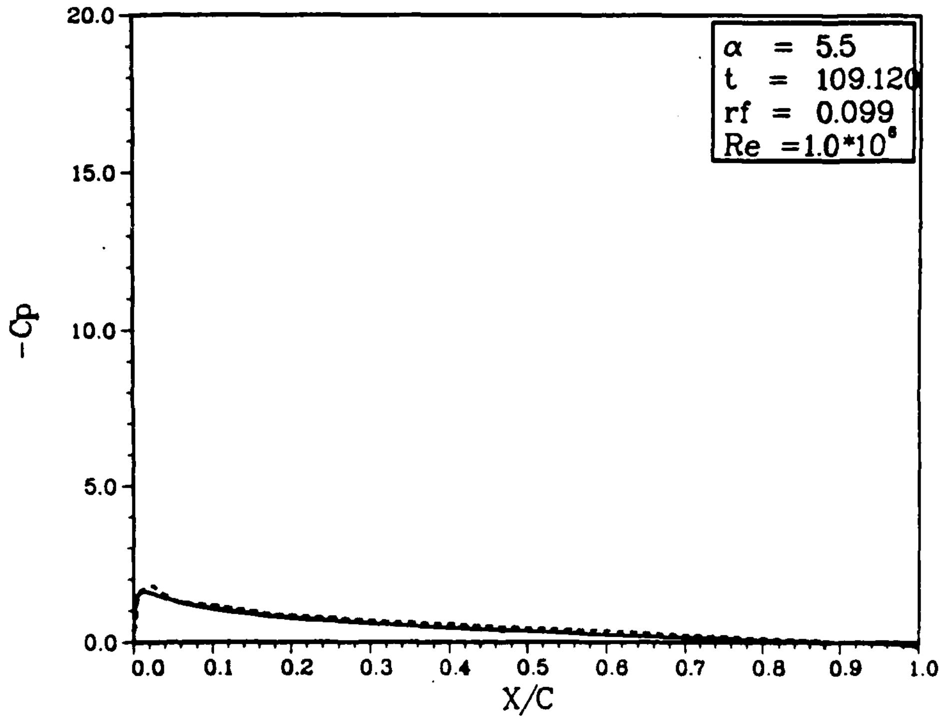
α = 14.5
t = 87.120
rf = 0.099
Re = 1.0*10⁶
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0


α = 11.4
t = 92.880
rf = 0.099
Re = 1.0*10^6
-Cp
X/C

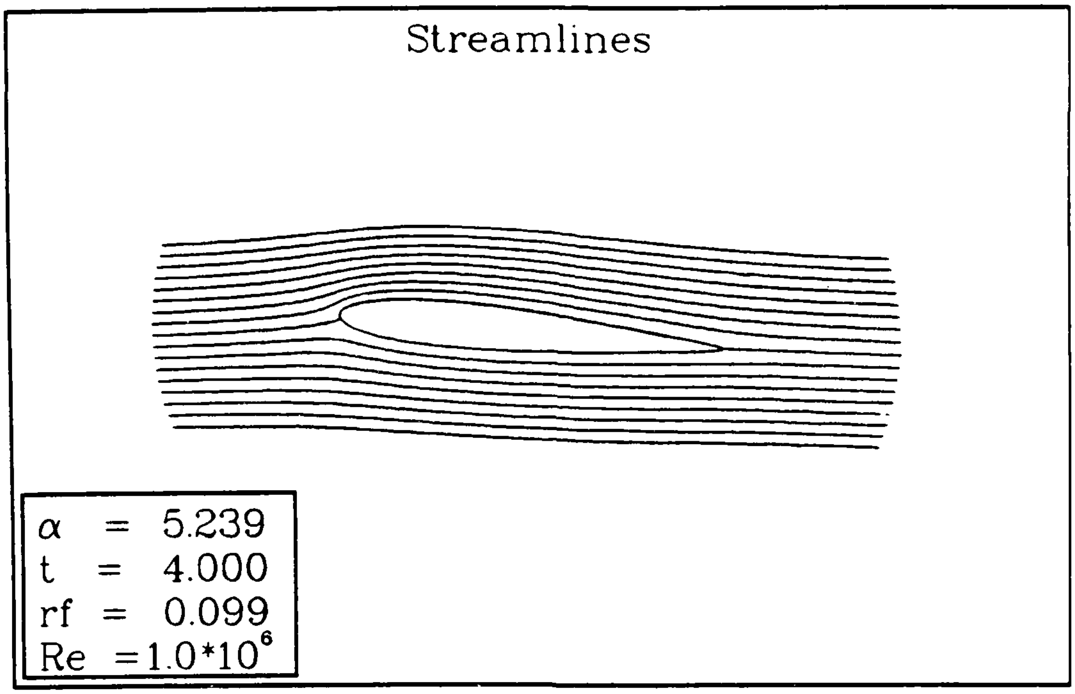

α = 8.7
t = 98.640
rf = 0.099
Re = 1.0*10^6
-Cp
X/C

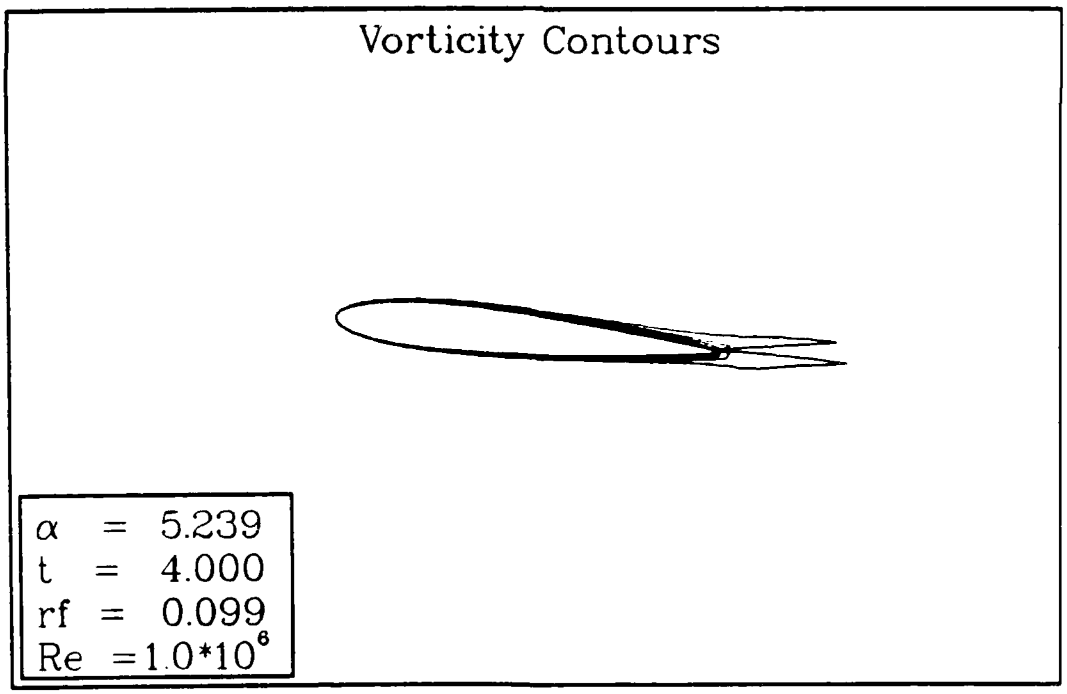


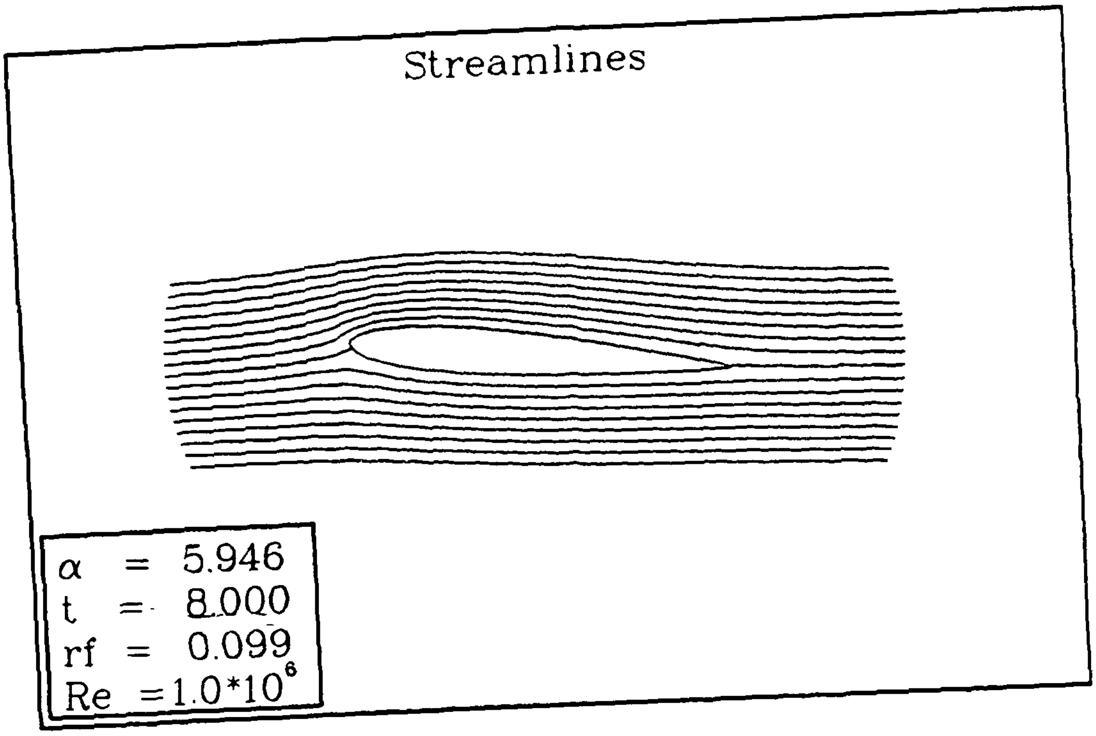
α = 6.7
t = 104.080
rf = 0.099
Re =1.0*10^6
-Cp
X/C

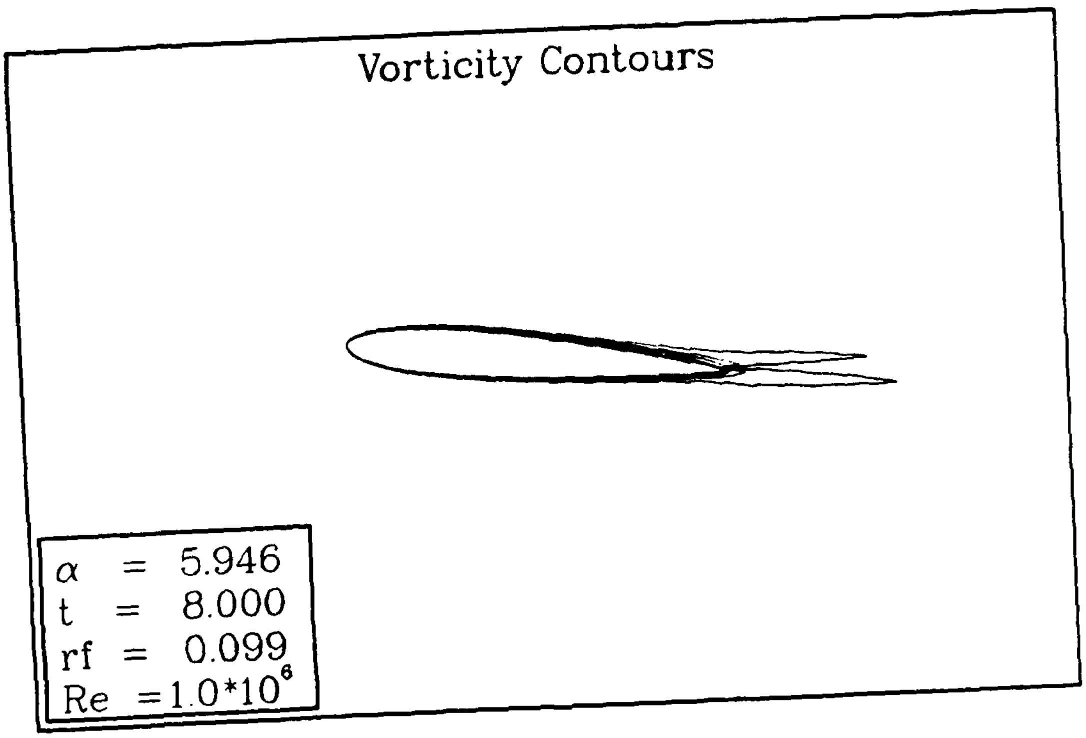


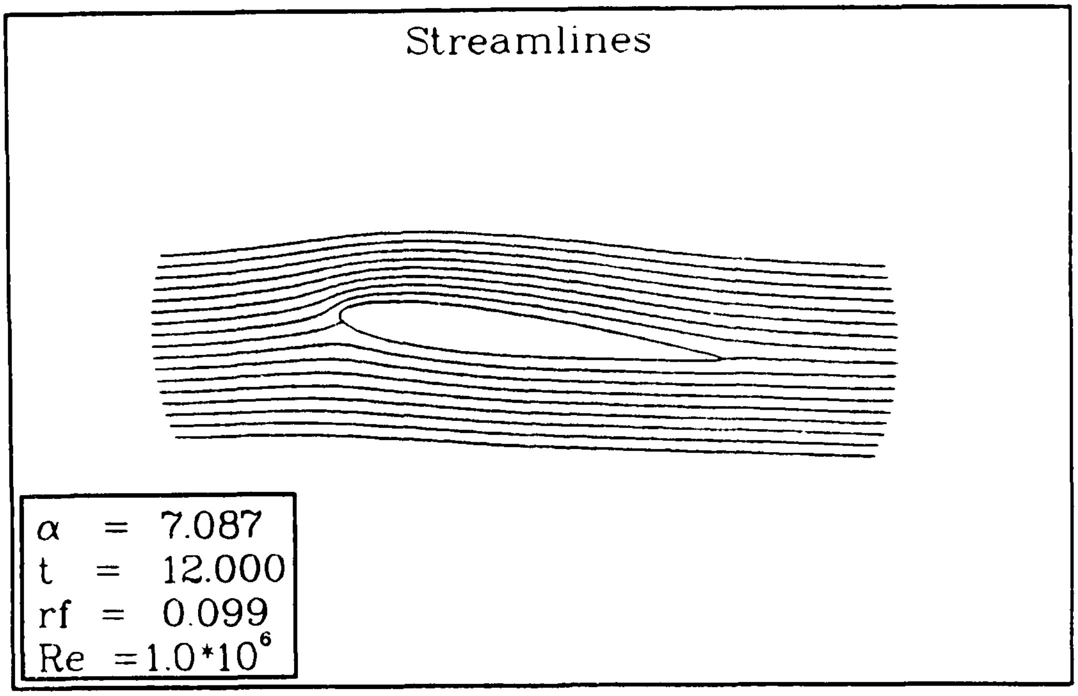
α = 5.5
t = 109.120
rf = 0.099
Re =1.0*10^6
-Cp
X/C

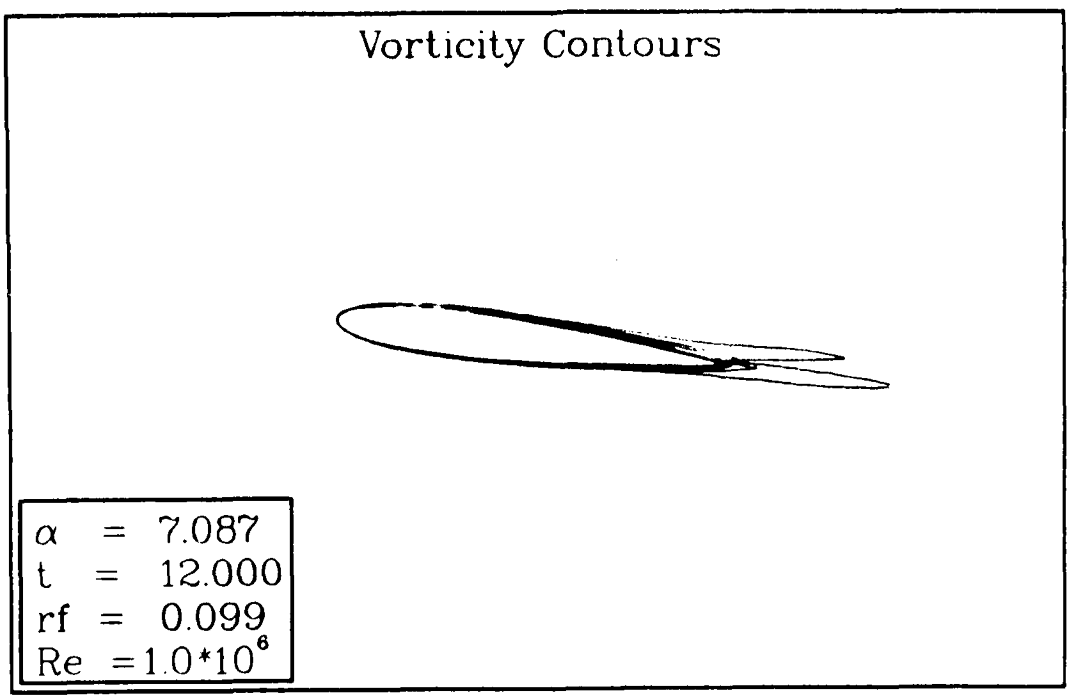

Streamlines
α = 5.239
t = 4.000
rf = 0.099
Re =1.0*10^6

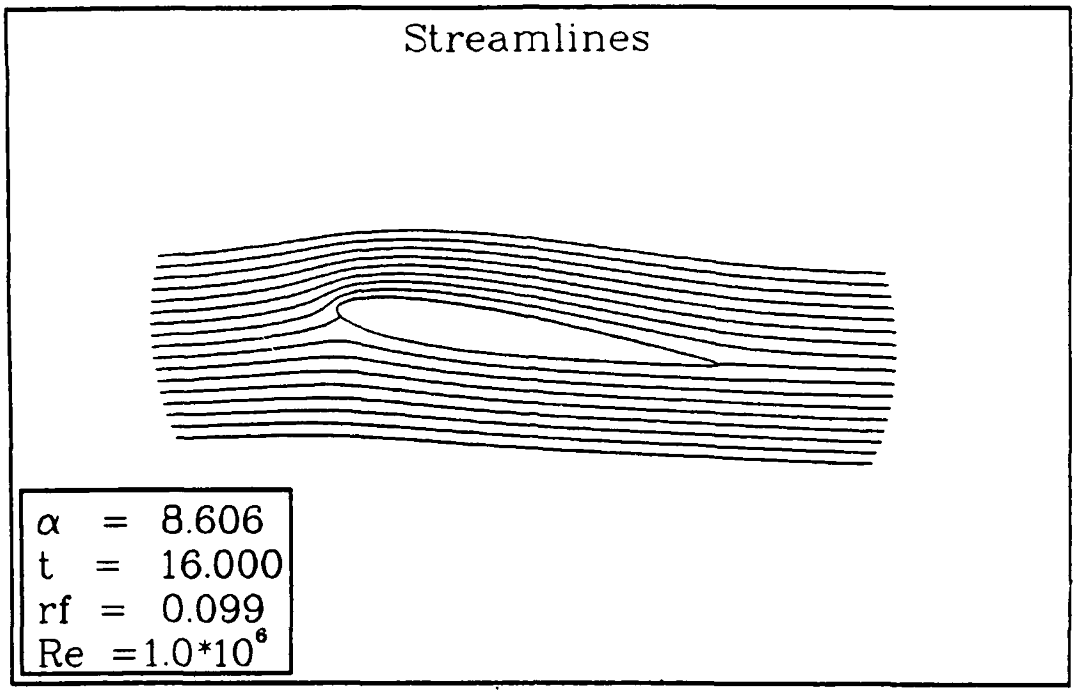

Vorticity Contours
α = 5.239
t = 4.000
rf = 0.099
Re =1.0*10^6

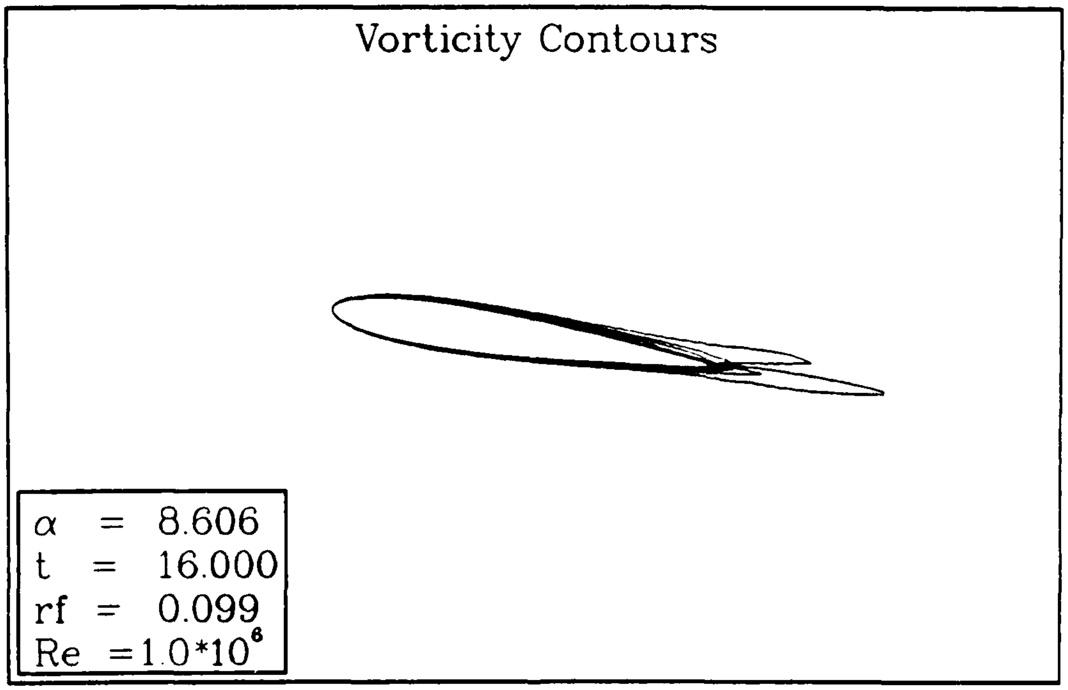

Streamlines
α = 5.946
t = 8.000
rf = 0.099
Re = 1.0*10^6

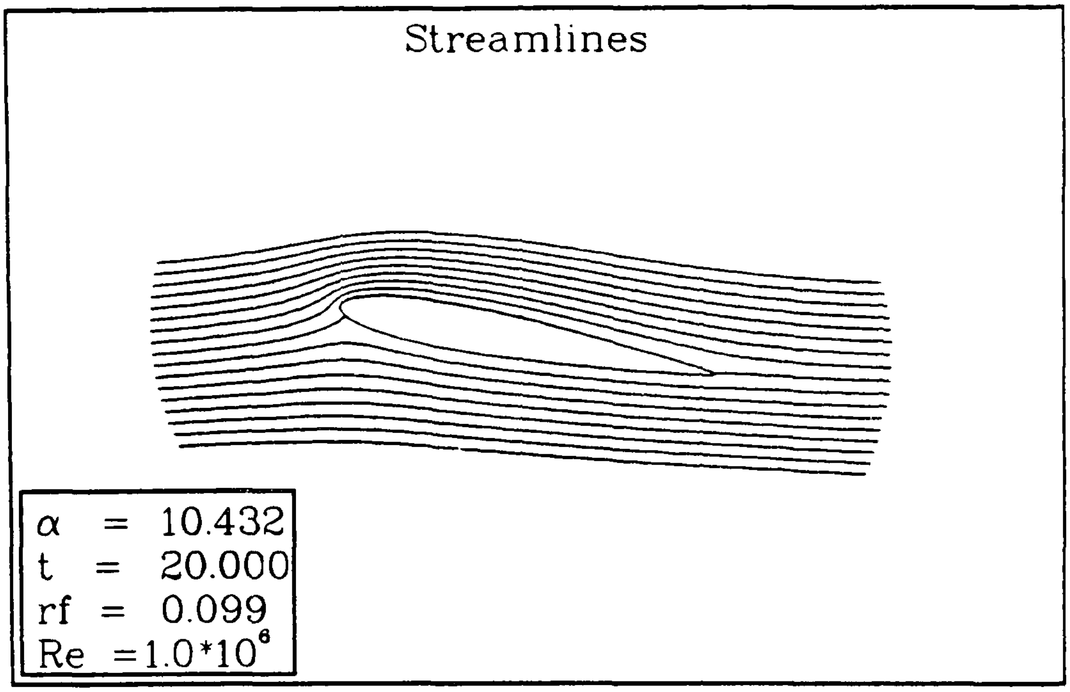

Vorticity Contours
α = 5.946
t = 8.000
rf = 0.099
Re = 1.0*10^6

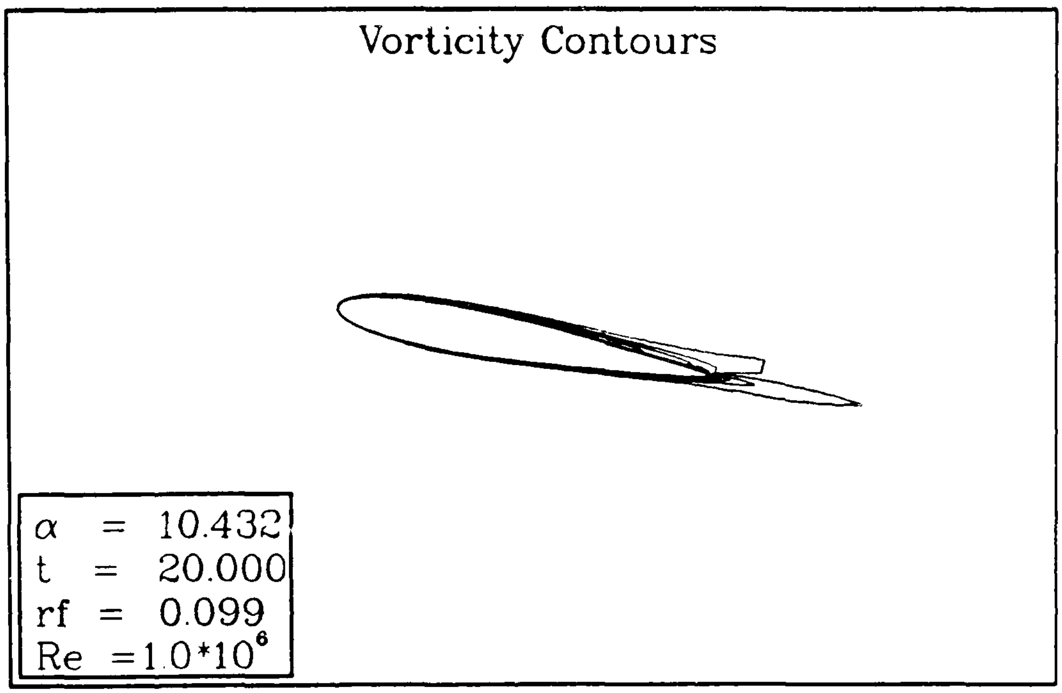

Streamlines
α = 7.087
t = 12.000
rf = 0.099
Re =1.0*10^6

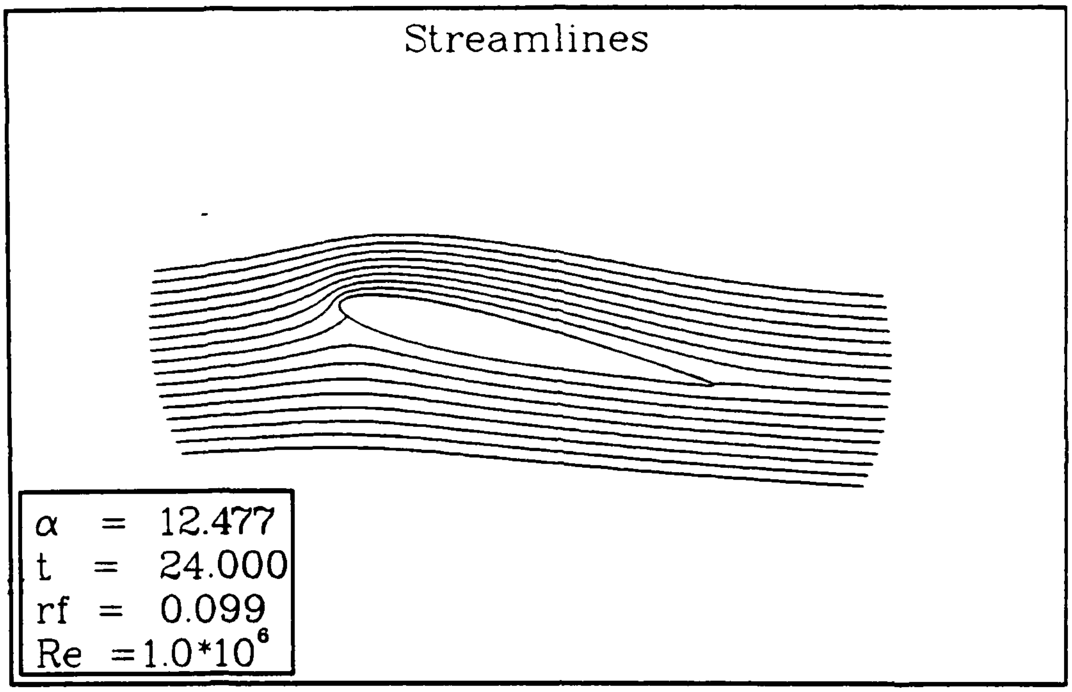

Vorticity Contours
α = 7.087
t = 12.000
rf = 0.099
Re =1.0*10^6

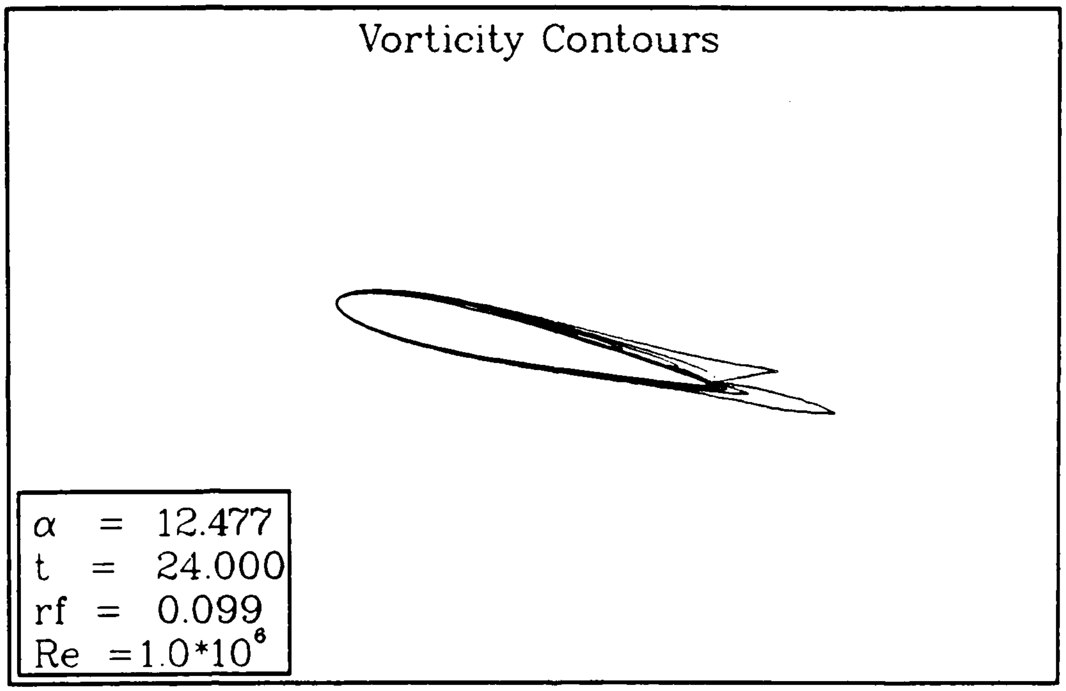

Streamlines
α = 8.606
t = 16.000
rf = 0.099
Re = 1.0*10^6

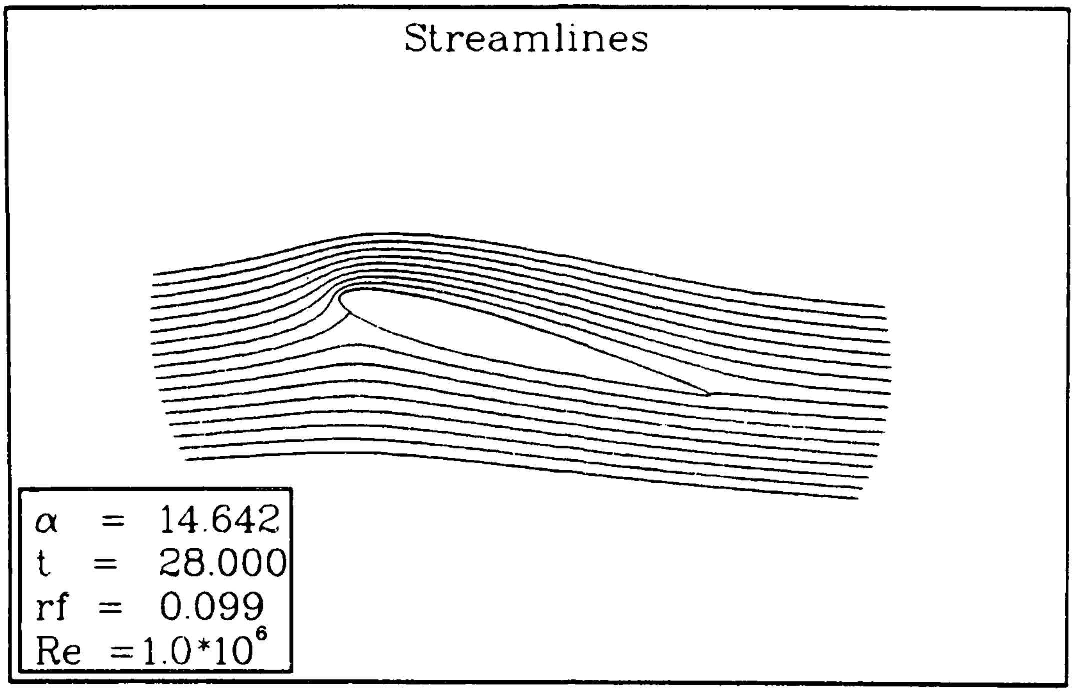

Vorticity Contours
α = 8.606
t = 16.000
rf = 0.099
Re = 1.0*10^6

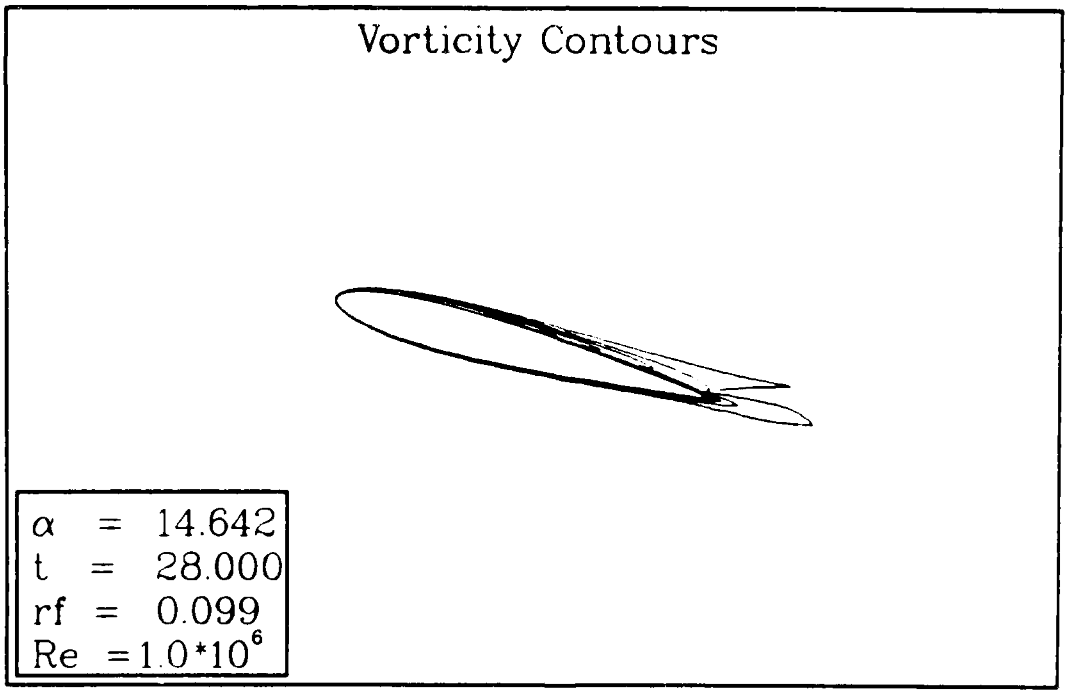

Streamlines
α = 10.432
t = 20.000
rf = 0.099
Re =1.0*10^6

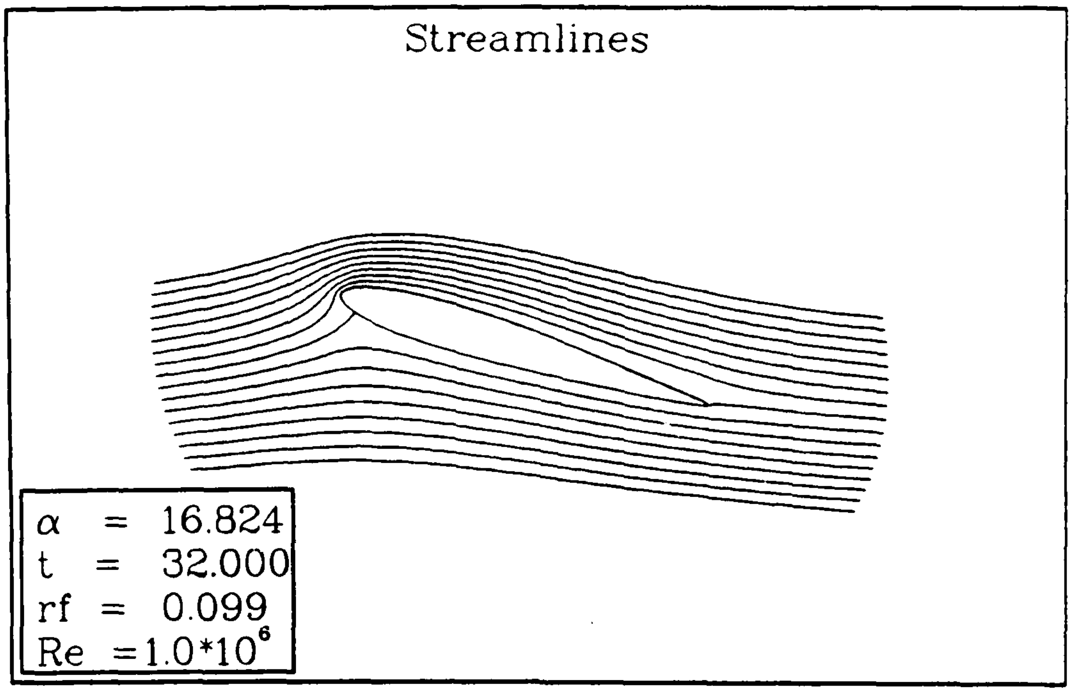

Vorticity Contours
α = 10.432
t = 20.000
rf = 0.099
Re =1.0*10^6

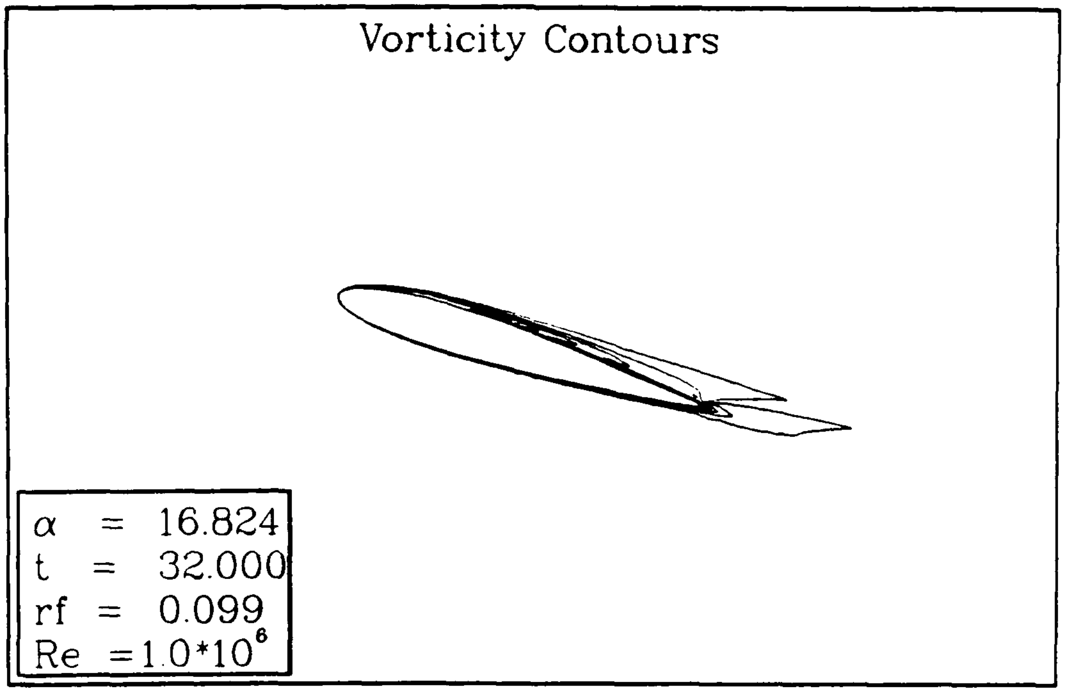

Streamlines
α = 12.477
t = 24.000
rf = 0.099
Re =1.0*10^6

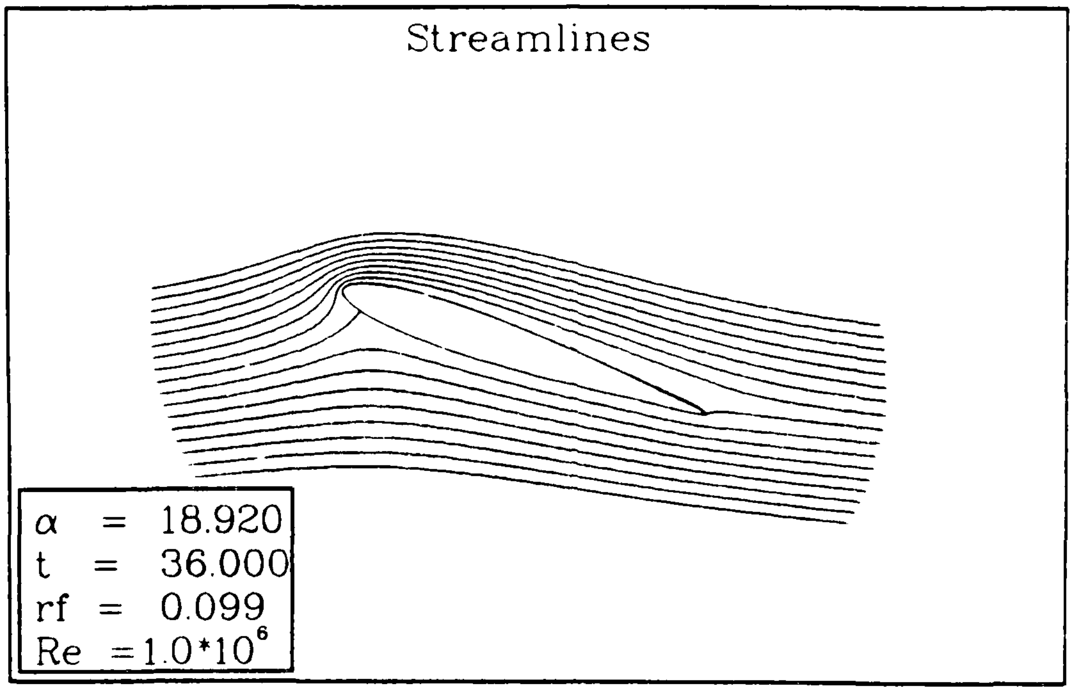

Vorticity Contours
α = 12.477
t = 24.000
rf = 0.099
Re =1.0*10^6

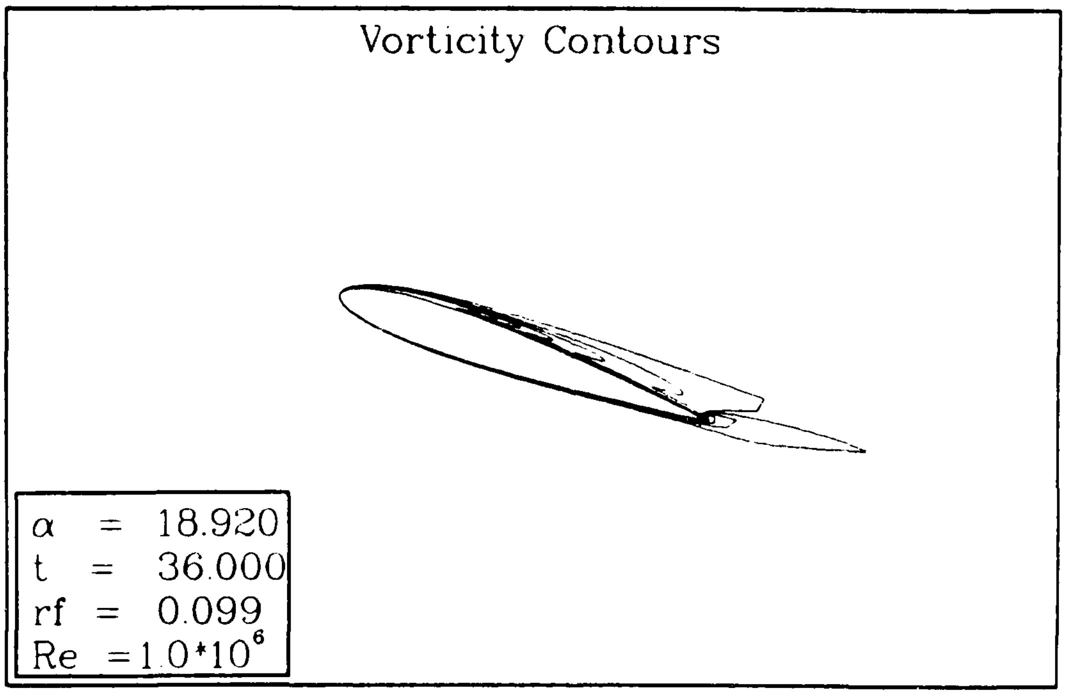

Streamlines
α = 14.642
t = 28.000
rf = 0.099
Re = $1.0*10^6$

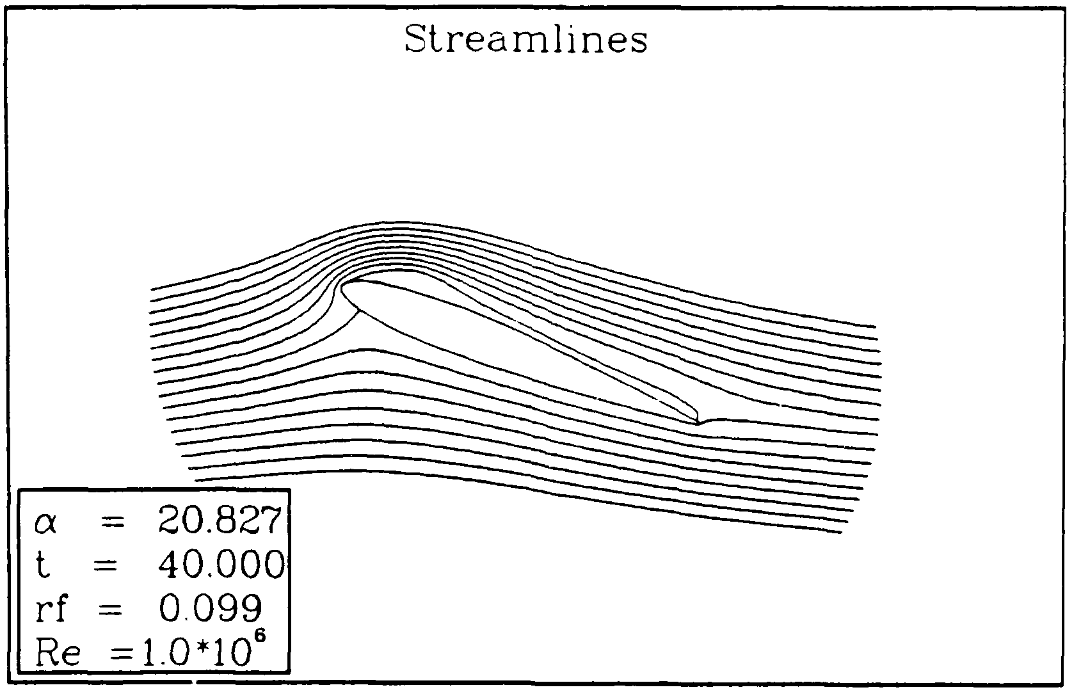

Vorticity Contours
α = 14.642
t = 28.000
rf = 0.099
Re = $1.0*10^6$

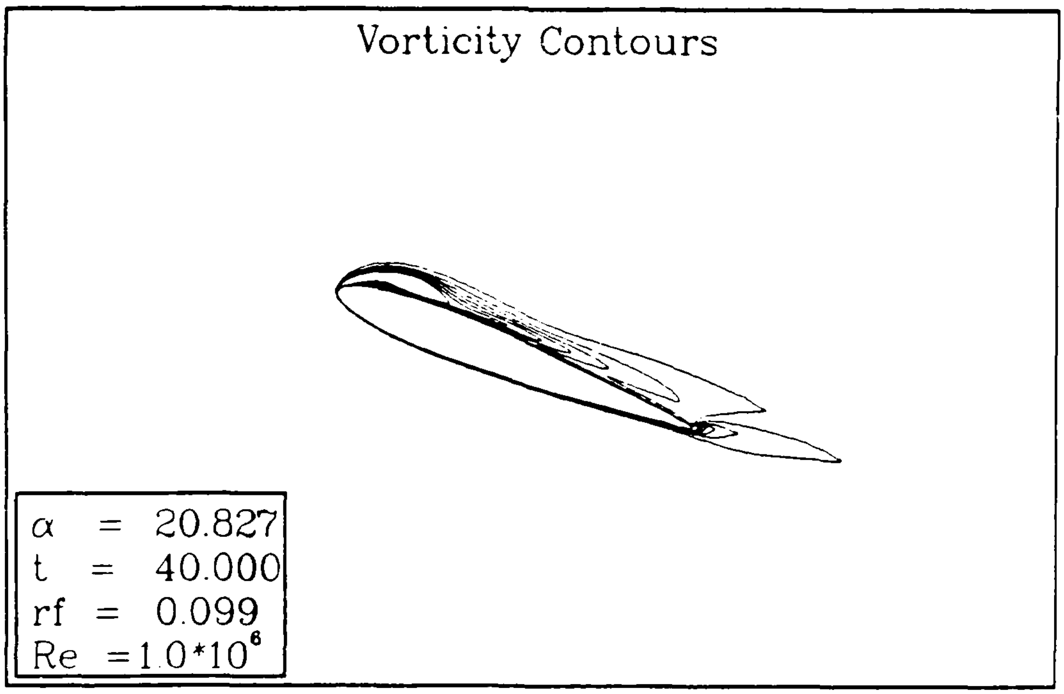

Streamlines
α = 16.824
t = 32.000
rf = 0.099
Re =1.0*10^6

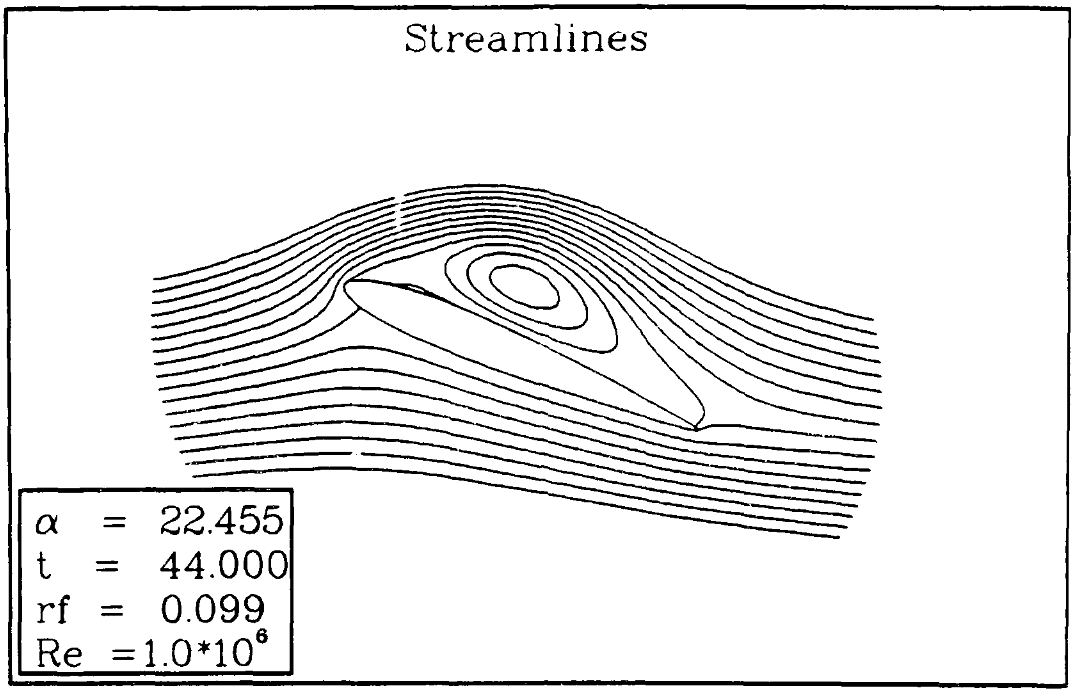

Vorticity Contours
α = 16.824
t = 32.000
rf = 0.099
Re =1.0*10^6

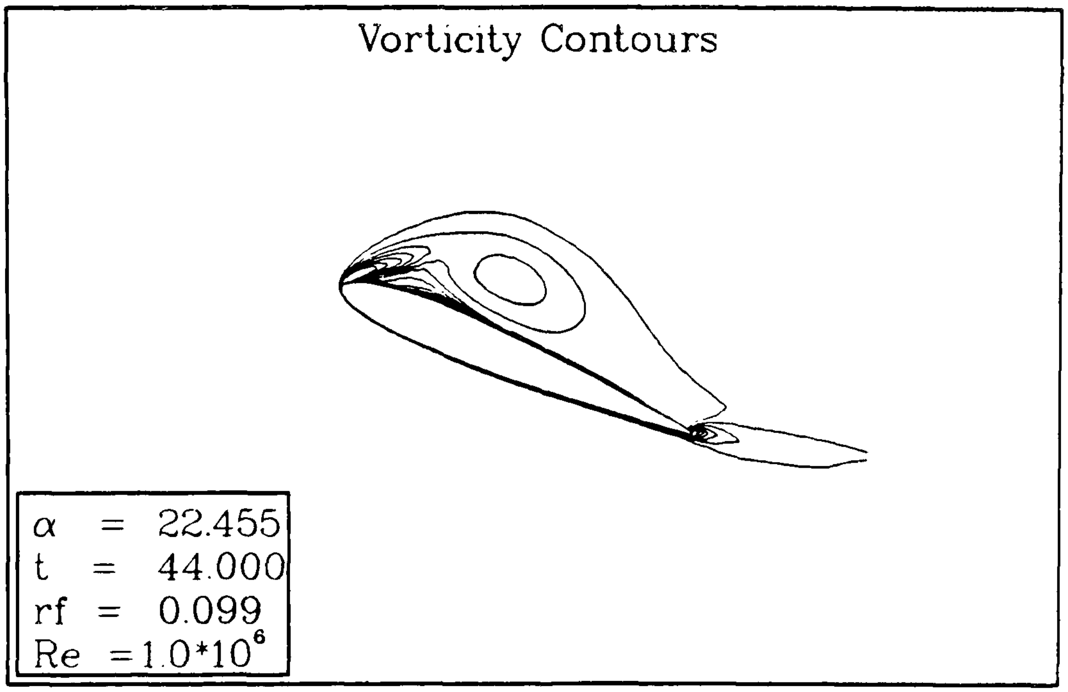

Streamlines
α = 18.920
t = 36.000
rf = 0.099
Re =1.0*10^6

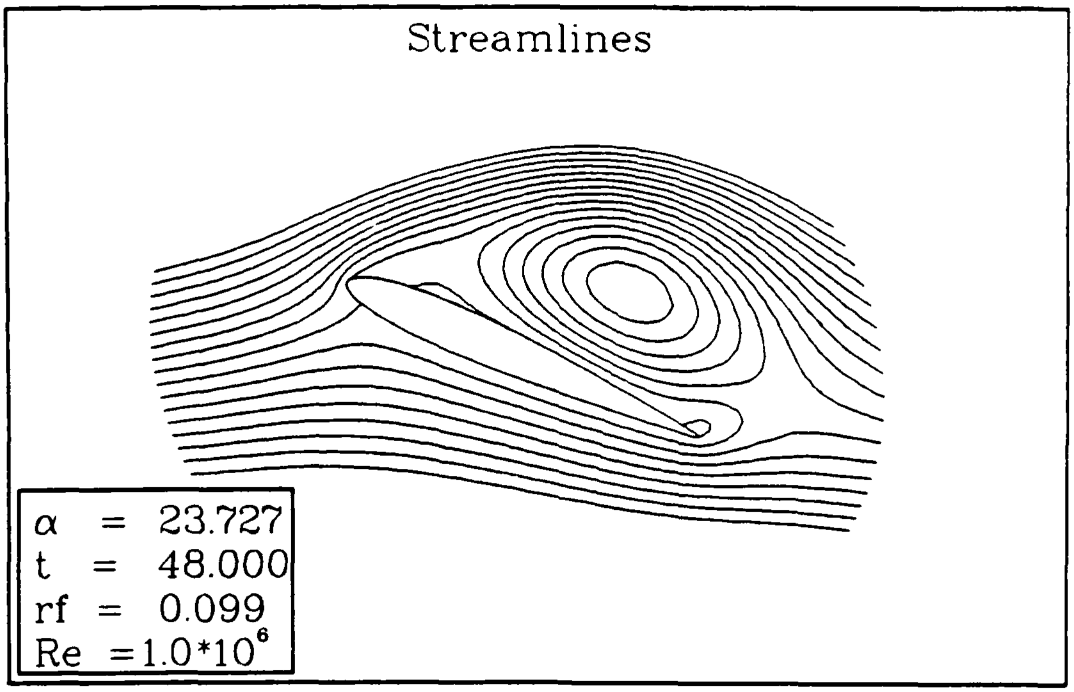

Vorticity Contours
α = 18.920
t = 36.000
rf = 0.099
Re =1.0*10^6

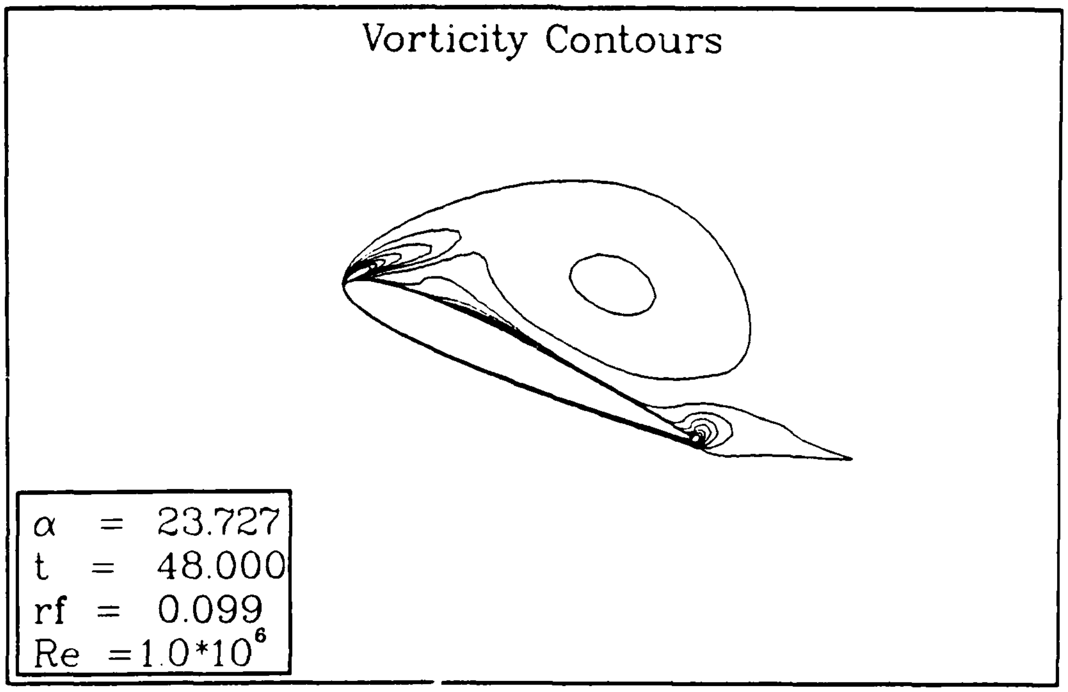

Streamlines
α = 20.827
t = 40.000
rf = 0.099
Re =1.0*10^6

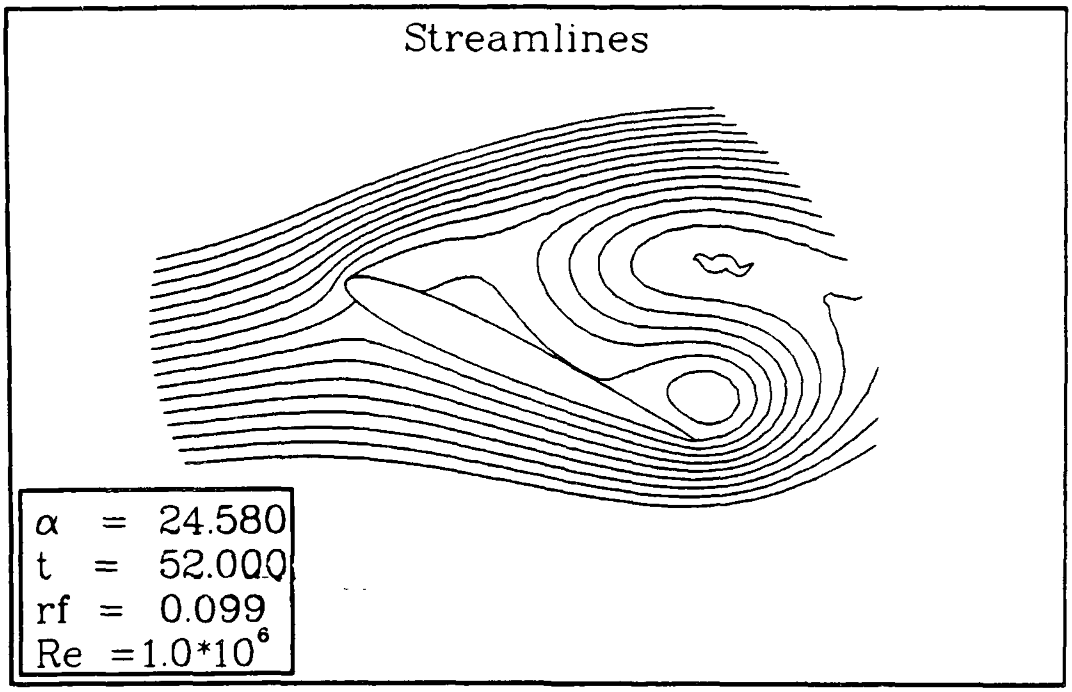

Vorticity Contours
α = 20.827
t = 40.000
rf = 0.099
Re =1.0*10^6

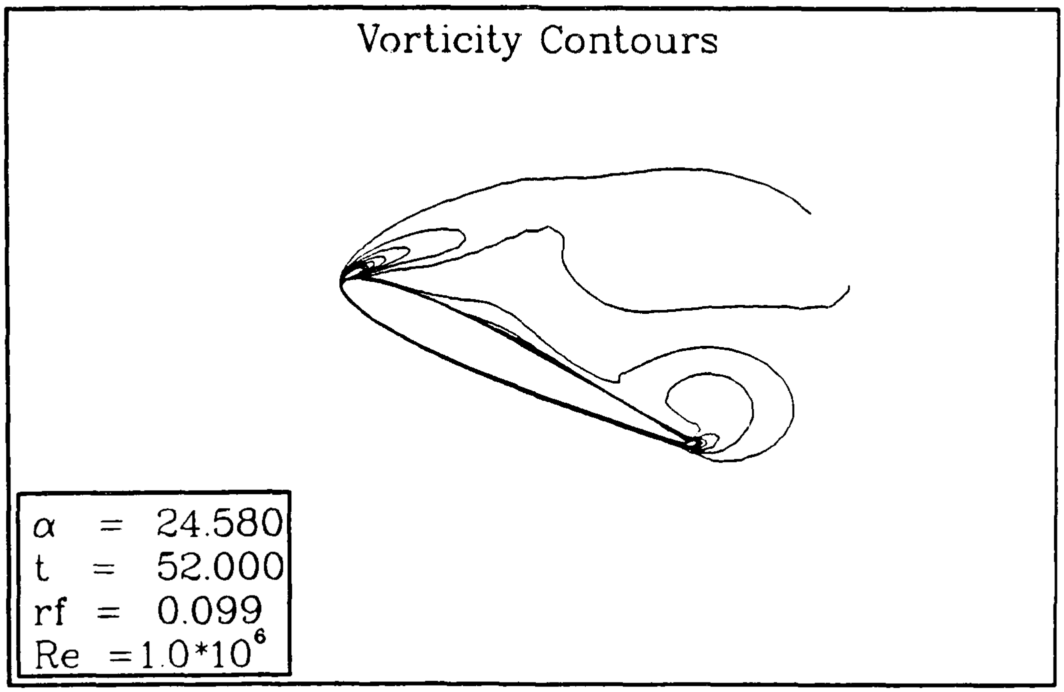

Streamlines
α = 22.455
t = 44.000
rf = 0.099
Re =1.0*10^6

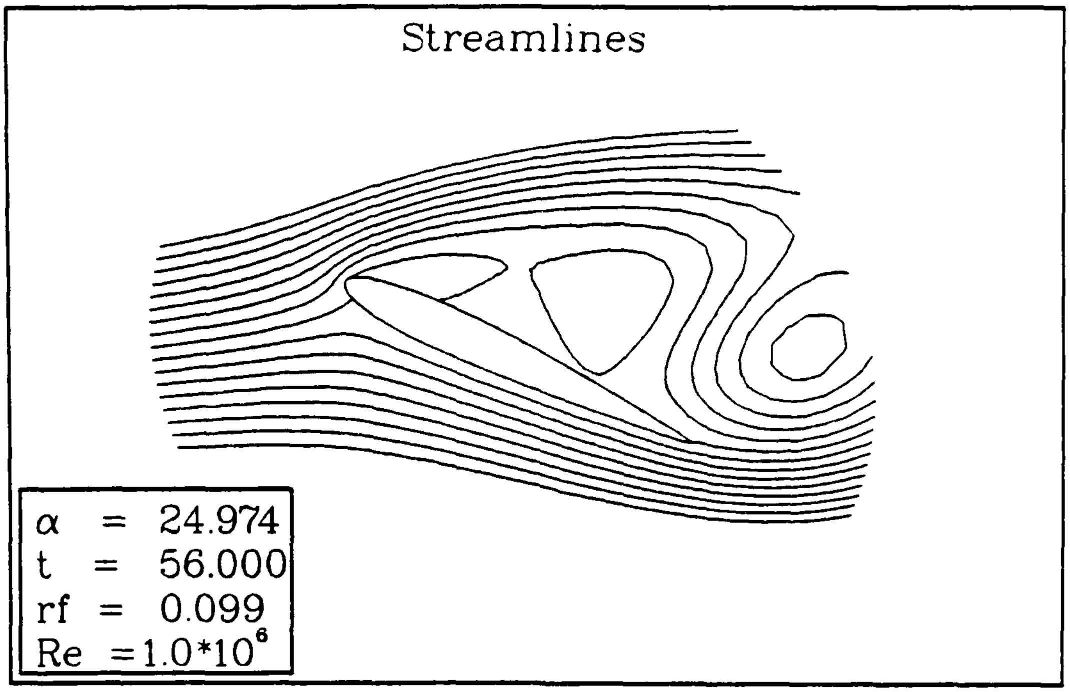

Vorticity Contours
α = 22.455
t = 44.000
rf = 0.099
Re =1.0*10^6

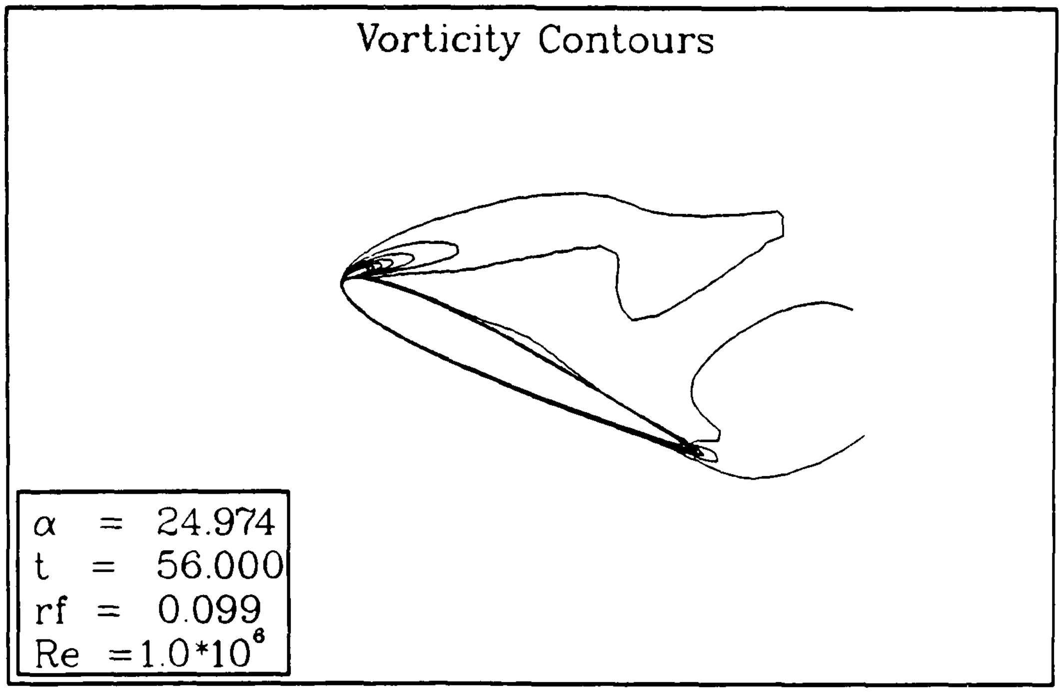

Streamlines
α = 23.727
t = 48.000
rf = 0.099
Re =1.0*10^6

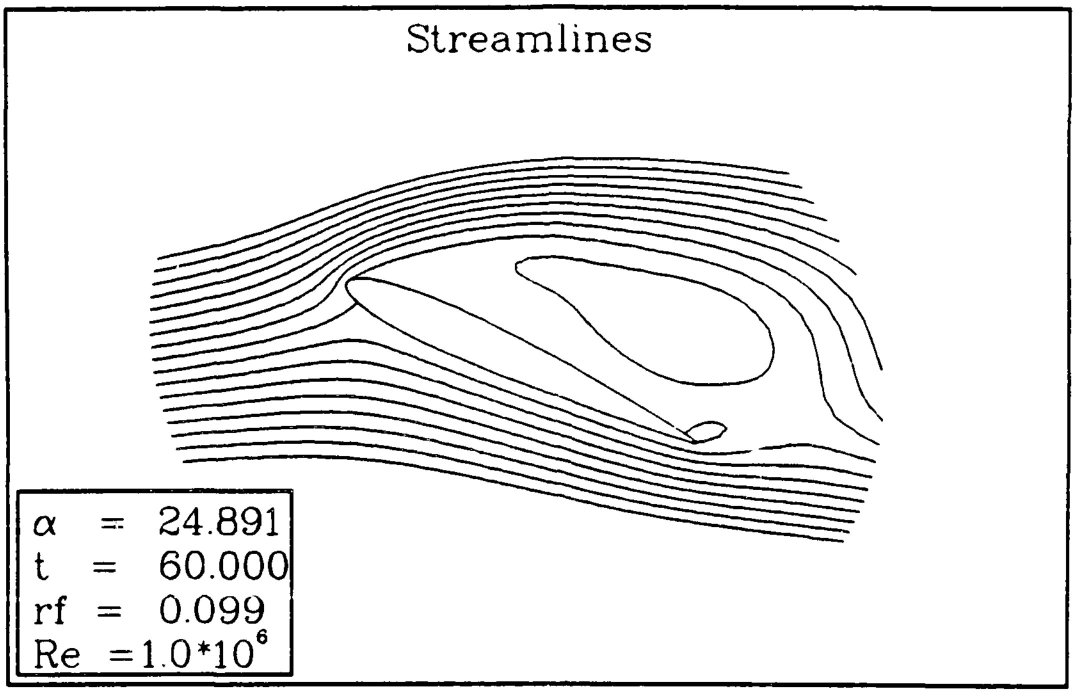

Vorticity Contours
α = 23.727
t = 48.000
rf = 0.099
Re =1.0*10^6

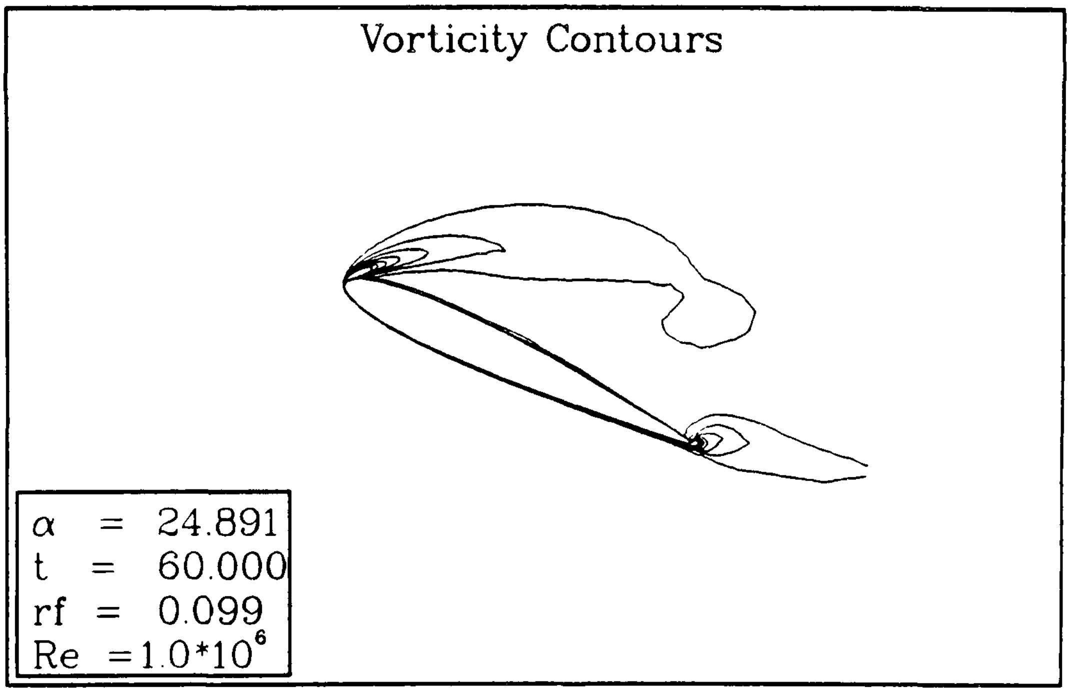

Streamlines
α = 24.580
t = 52.000
rf = 0.099
Re =1.0*10^6

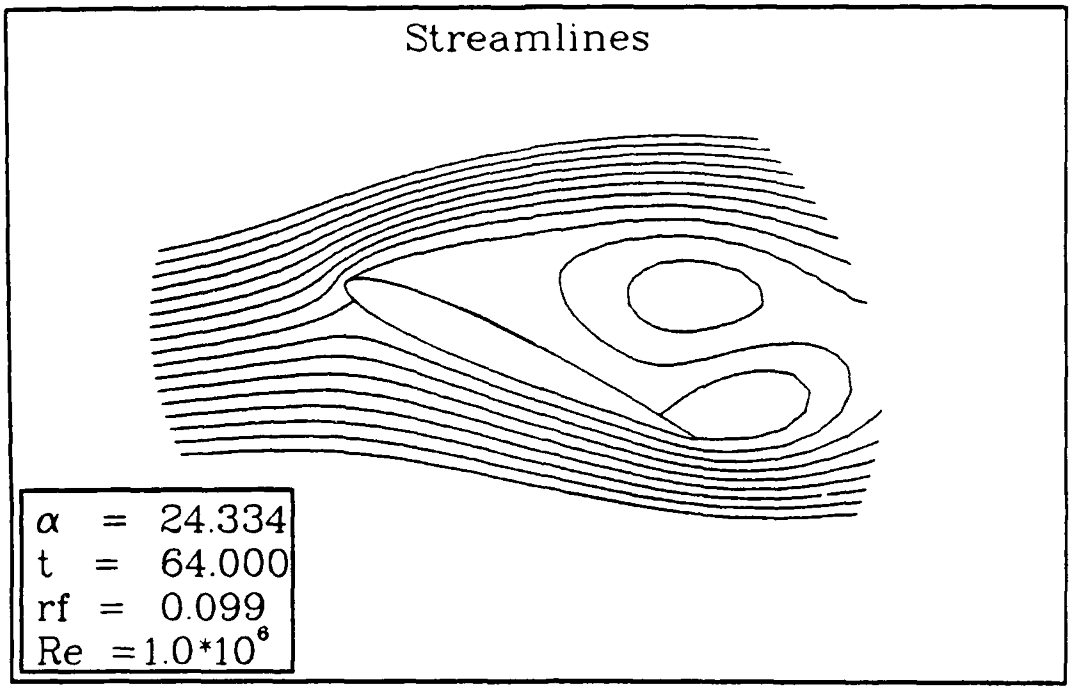

Vorticity Contours
α = 24.580
t = 52.000
rf = 0.099
Re =1.0*10^6

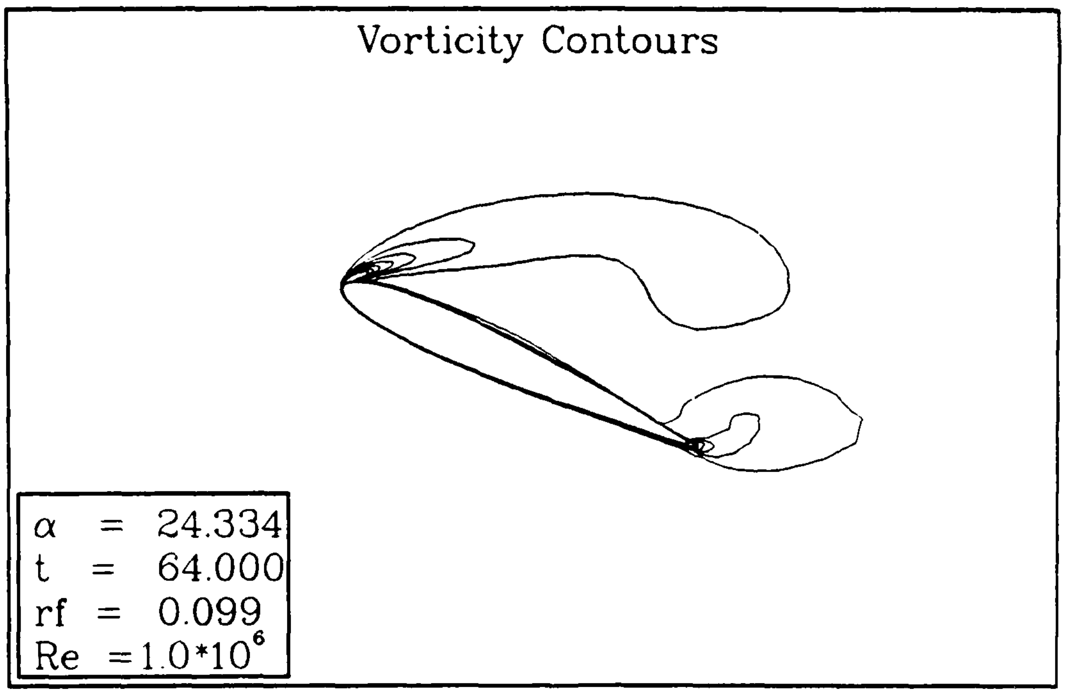

Streamlines
α = 24.974
t = 56.000
rf = 0.099
Re =1.0*10^6

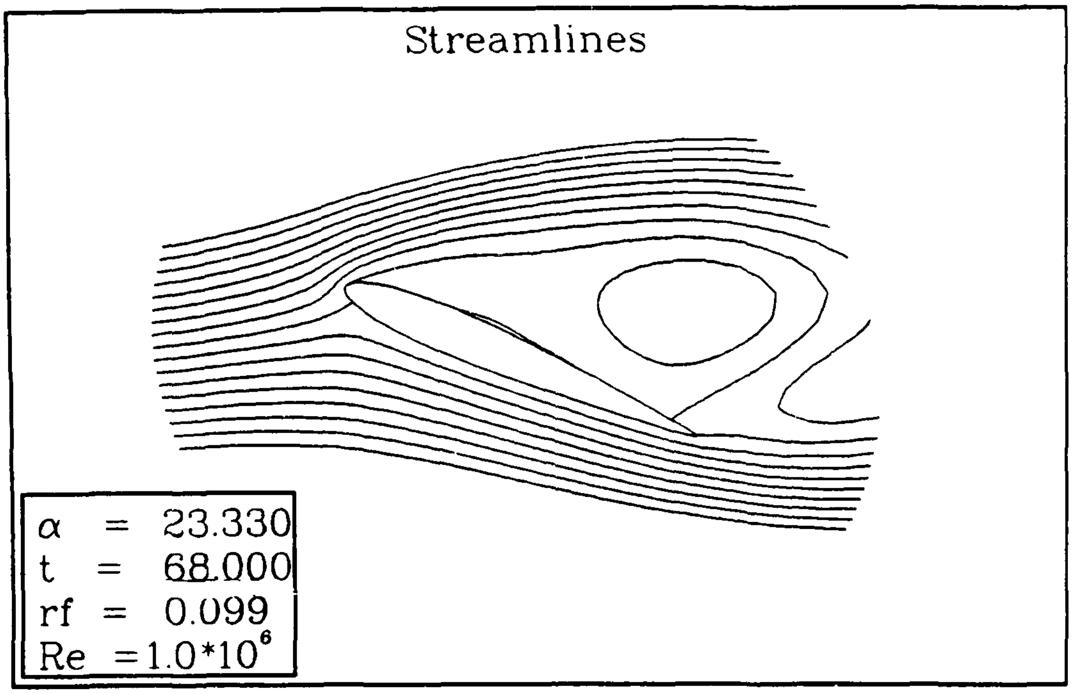

Vorticity Contours
α = 24.974
t = 56.000
rf = 0.099
Re =1.0*10^6

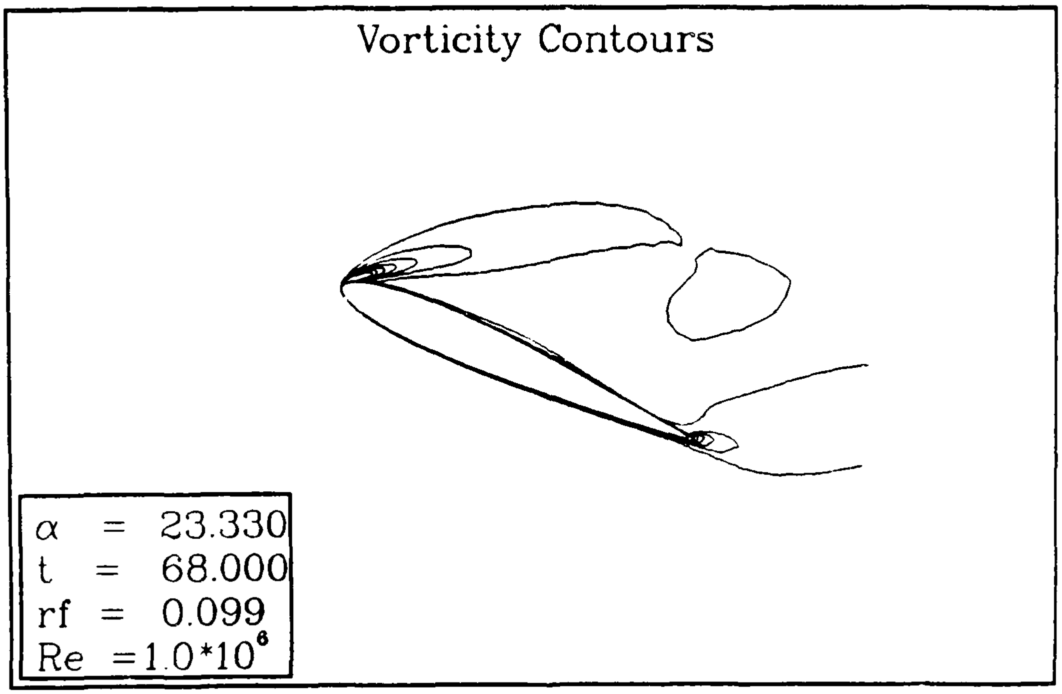

Streamlines
α = 24.891
t = 60.000
rf = 0.099
Re =1.0*10^6

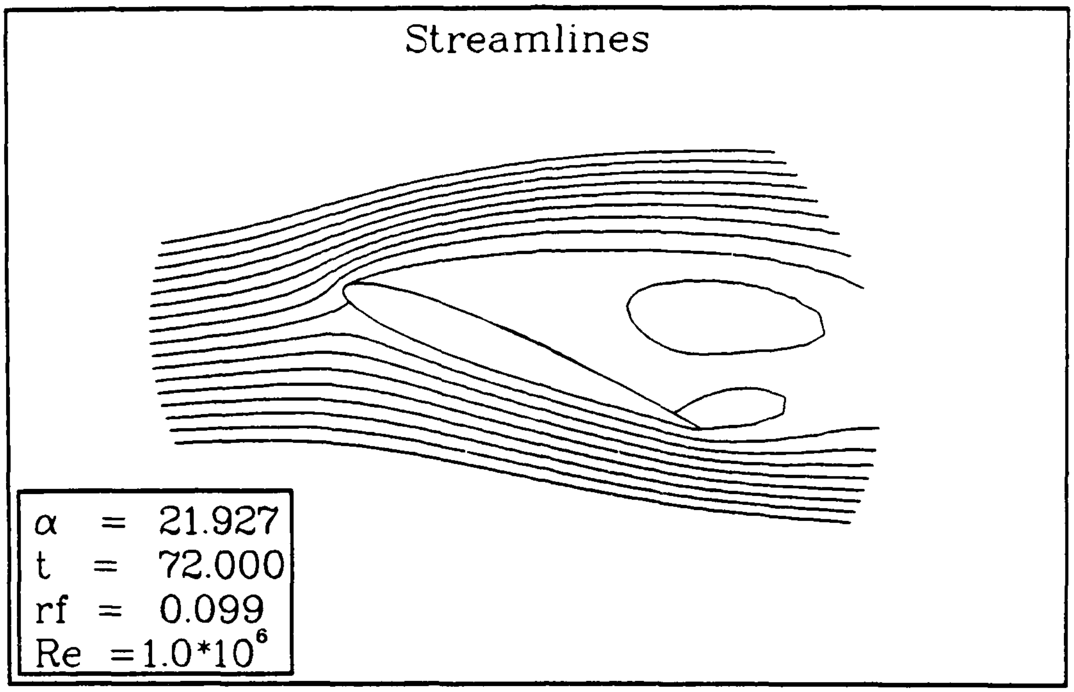

Vorticity Contours
α = 24.891
t = 60.000
rf = 0.099
Re =1.0*10^6

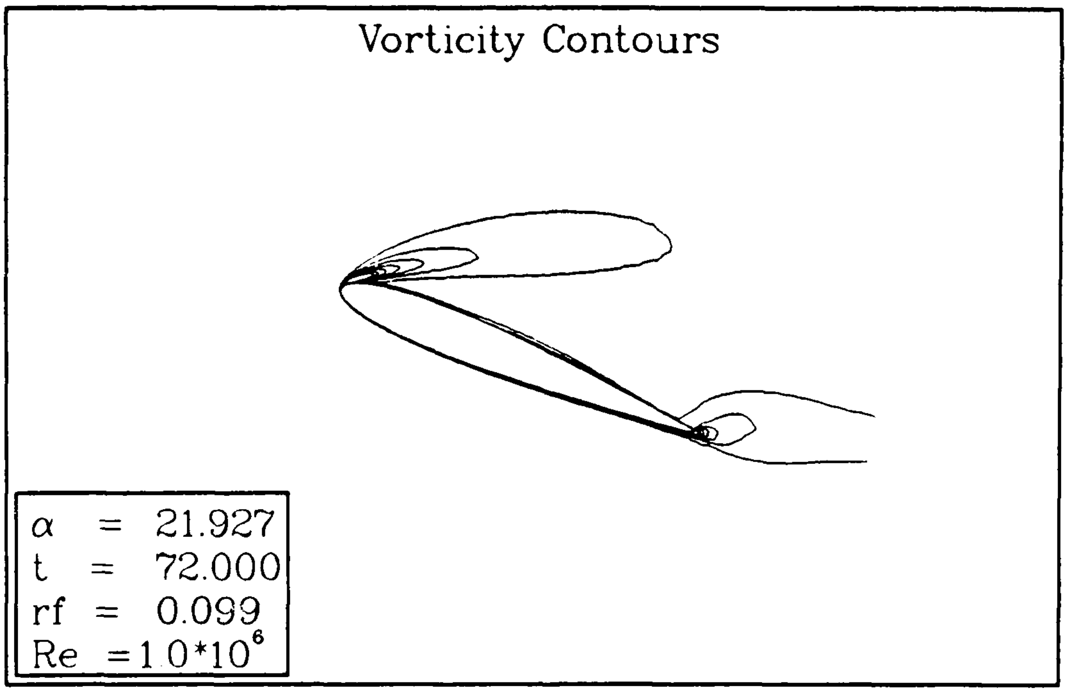

Streamlines
α = 24.334
t = 64.000
rf = 0.099
Re =1.0*10^6

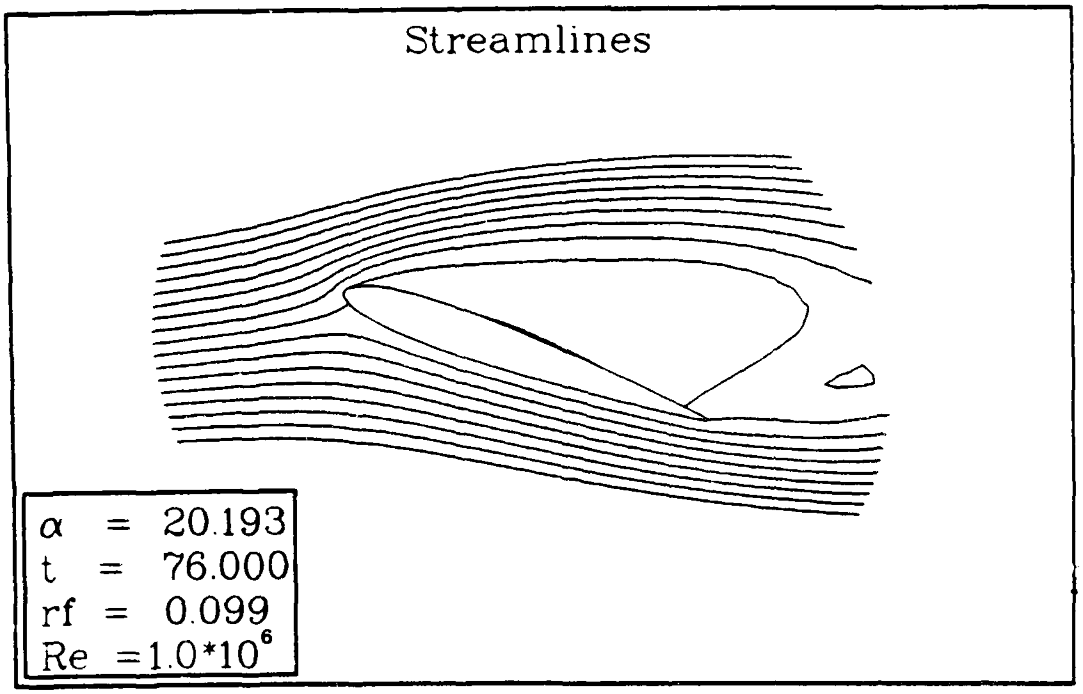

Vorticity Contours
α = 24.334
t = 64.000
rf = 0.099
Re =1.0*10^6

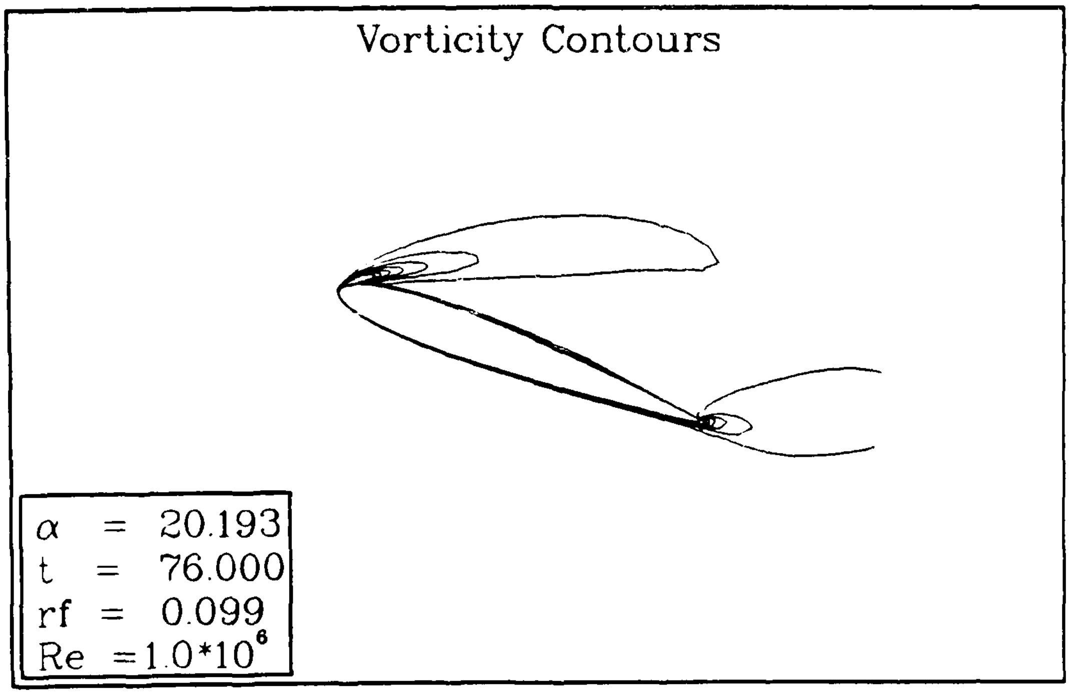

Streamlines
α = 23.330
t = 68.000
rf = 0.099
Re =1.0*10^6

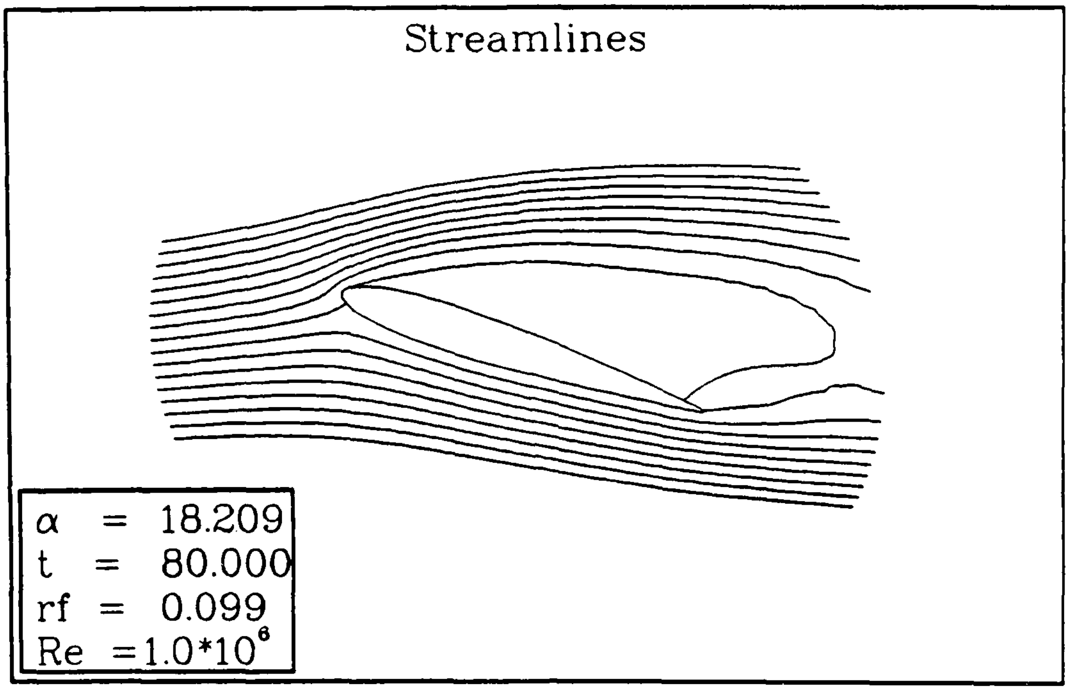

Vorticity Contours
α = 23.330
t = 68.000
rf = 0.099
Re =1.0*10^6

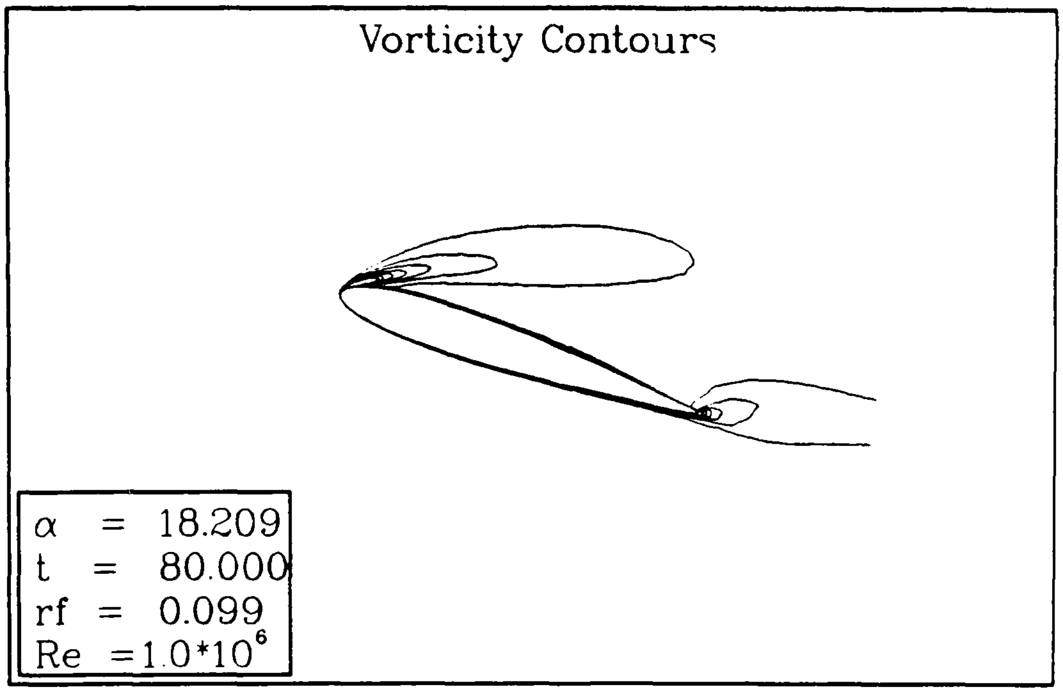

Streamlines
α = 21.927
t = 72.000
rf = 0.099
Re =1.0*10^6

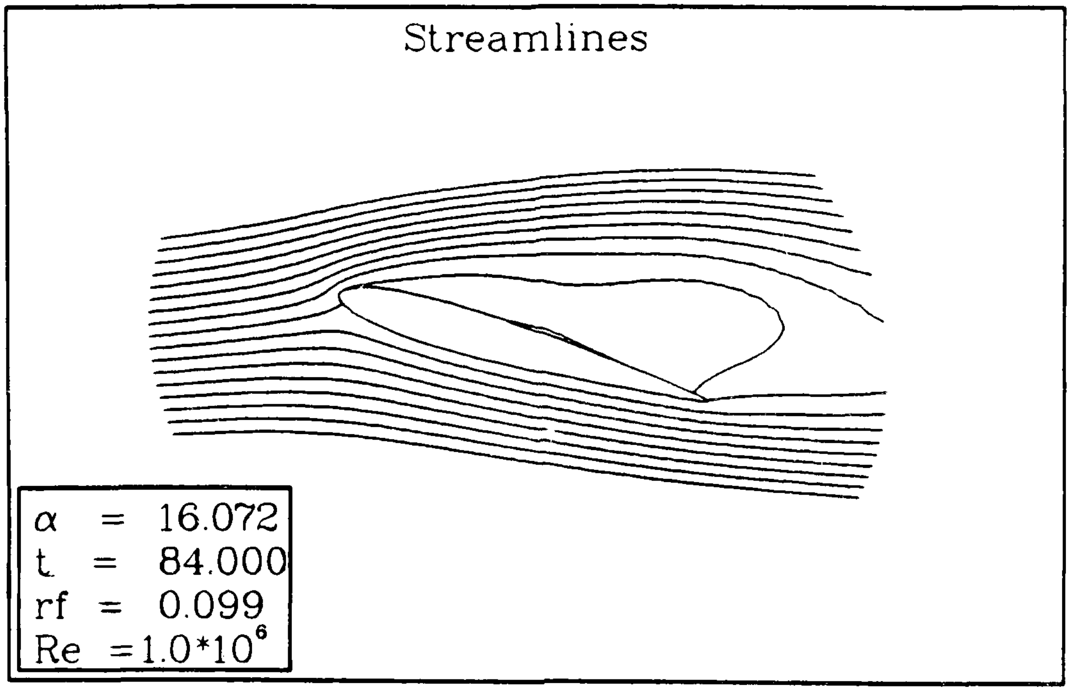

Vorticity Contours
α = 21.927
t = 72.000
rf = 0.099
Re =1.0*10^6

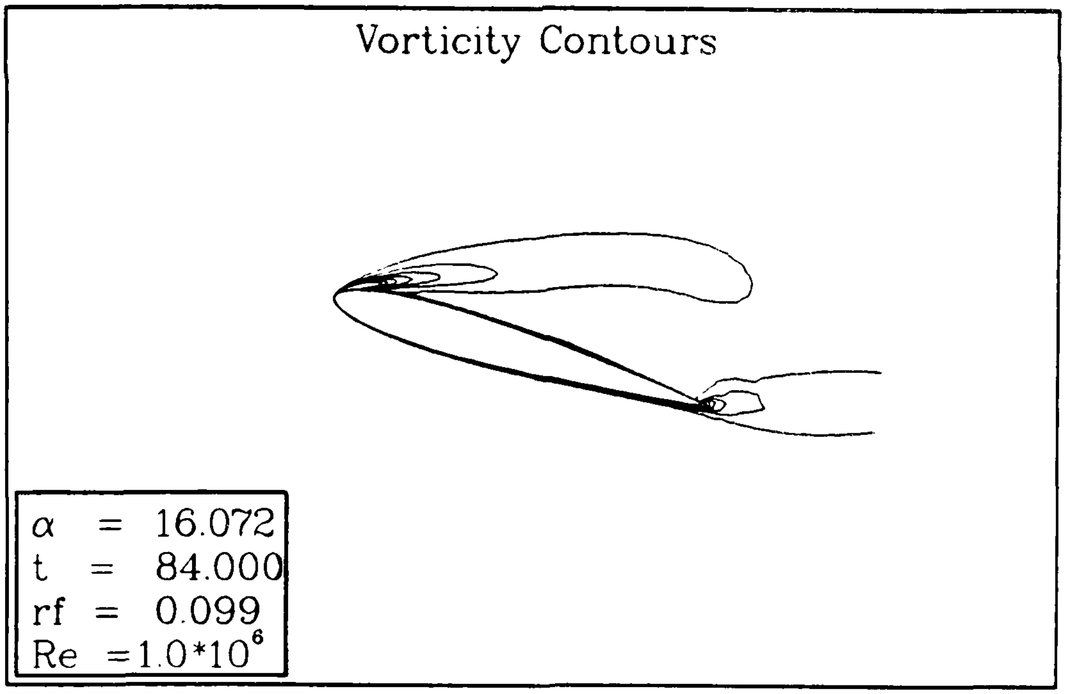

Streamlines
α = 20.193
t = 76.000
rf = 0.099
Re =1.0*10^6

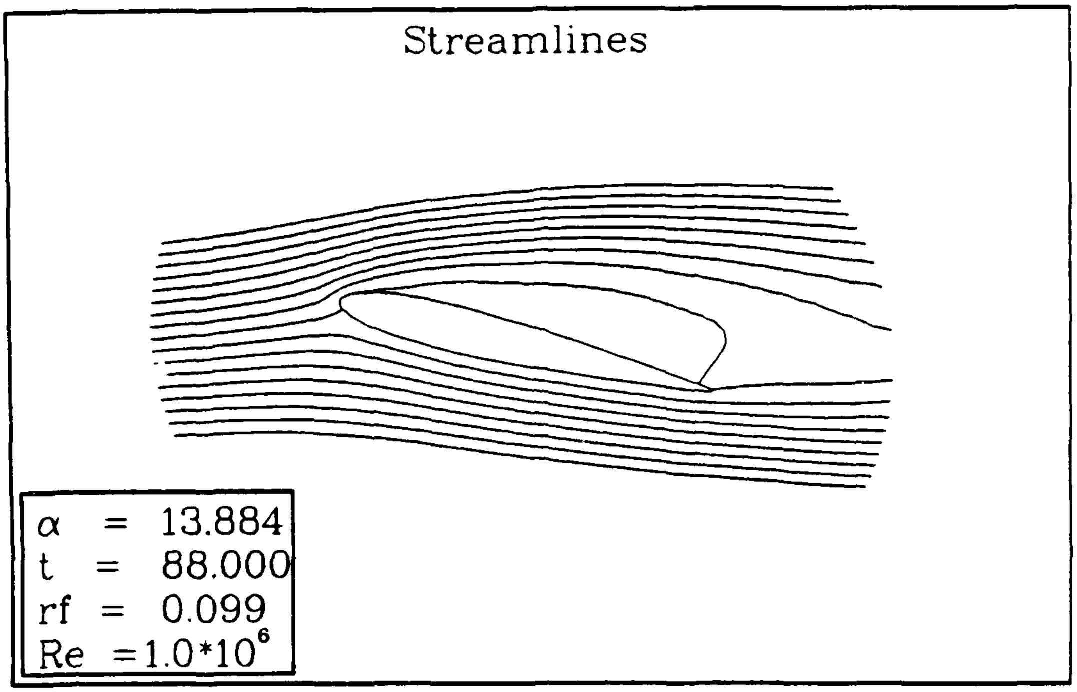

Vorticity Contours
α = 20.193
t = 76.000
rf = 0.099
Re =1.0*10^6

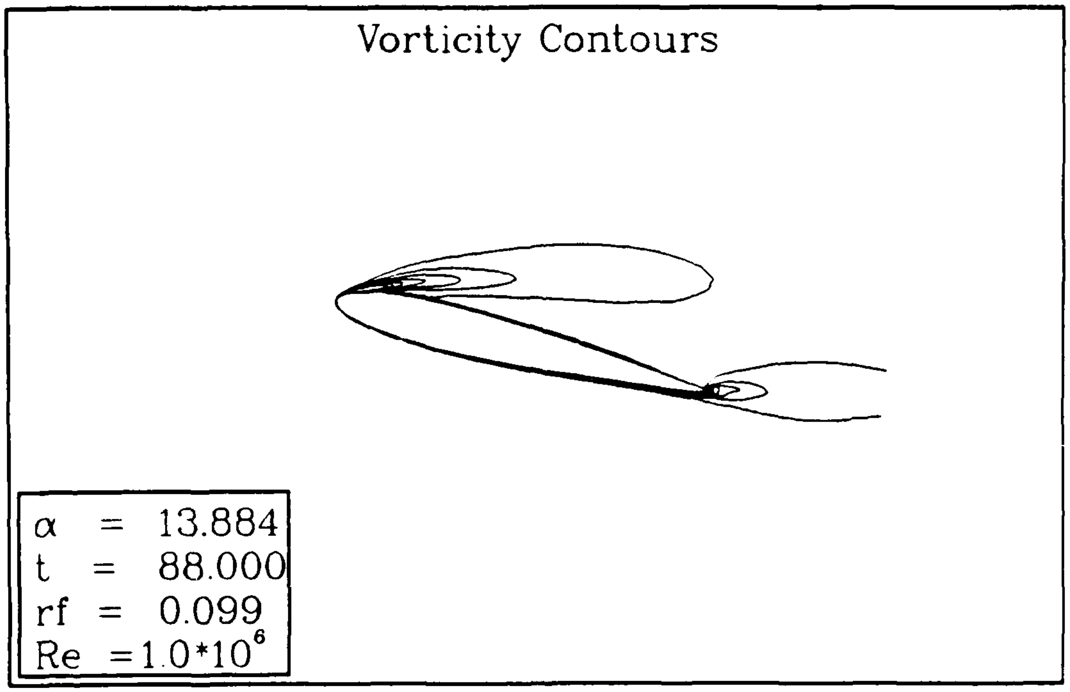

Streamlines
α = 18.209
t = 80.000
rf = 0.099
Re =1.0*10^6

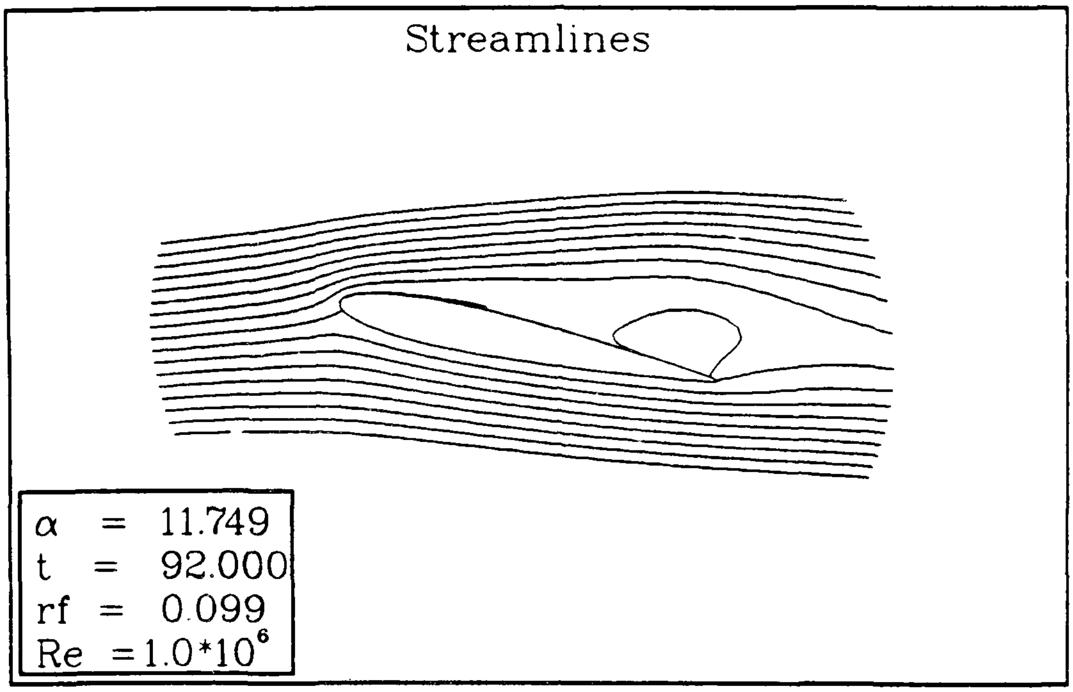

Vorticity Contours
α = 18.209
t = 80.000
rf = 0.099
Re =1.0*10^6

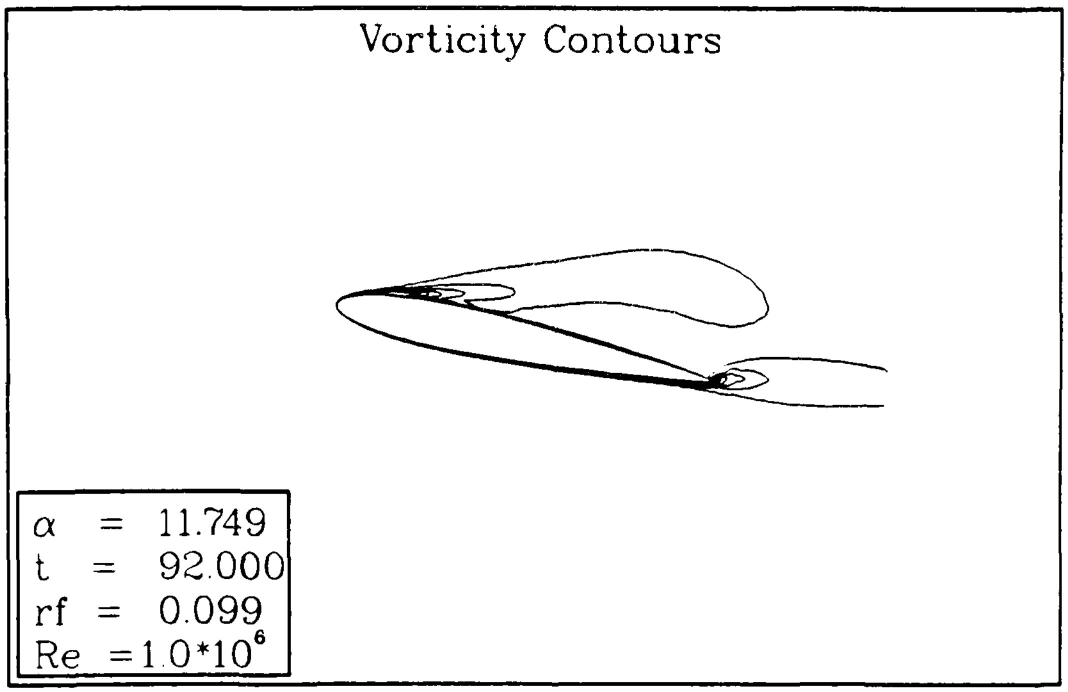

Streamlines
α = 16.072
t = 84.000
rf = 0.099
Re =1.0*10^6

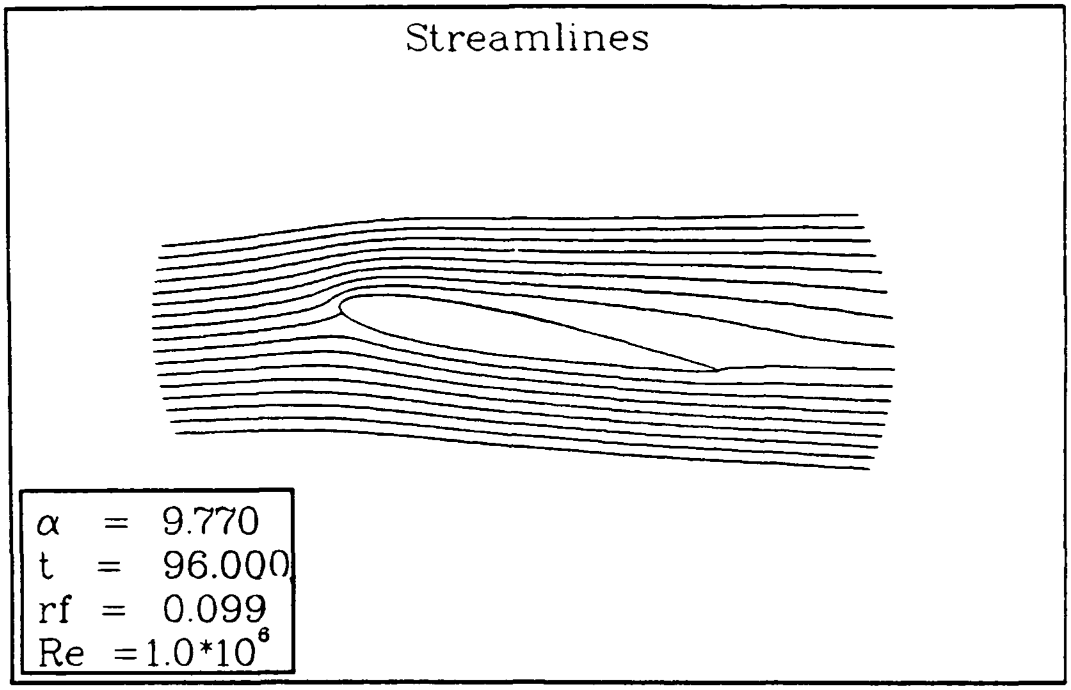

Vorticity Contours
α = 16.072
t = 84.000
rf = 0.099
Re =1.0*10^6

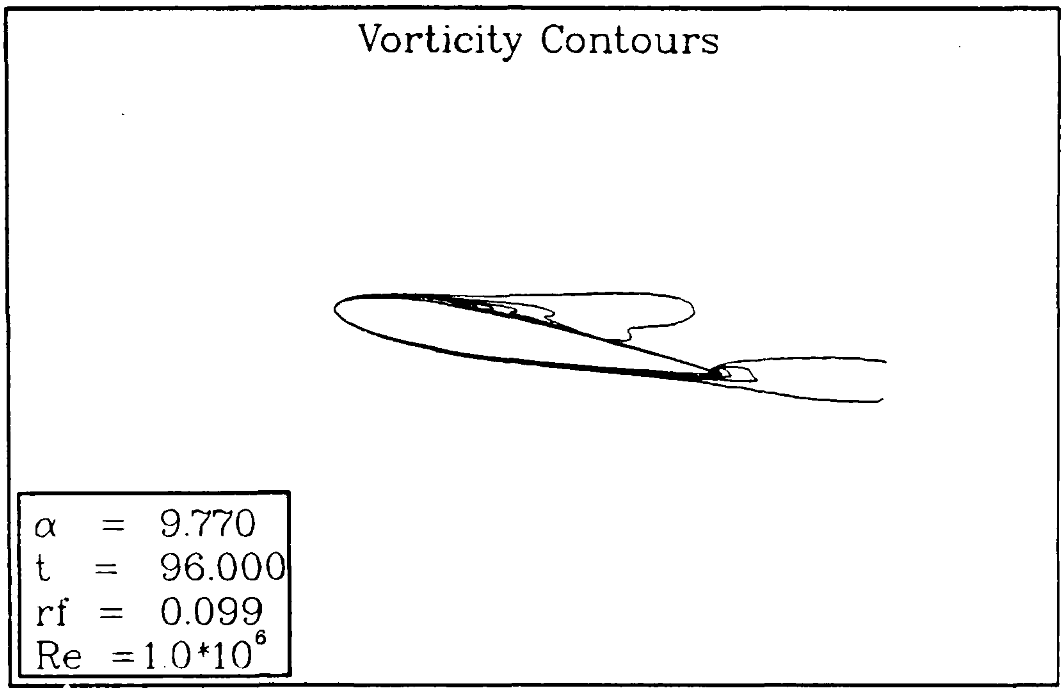

Streamlines
α = 13.884
t = 88.000
rf = 0.099
Re =1.0*10^6

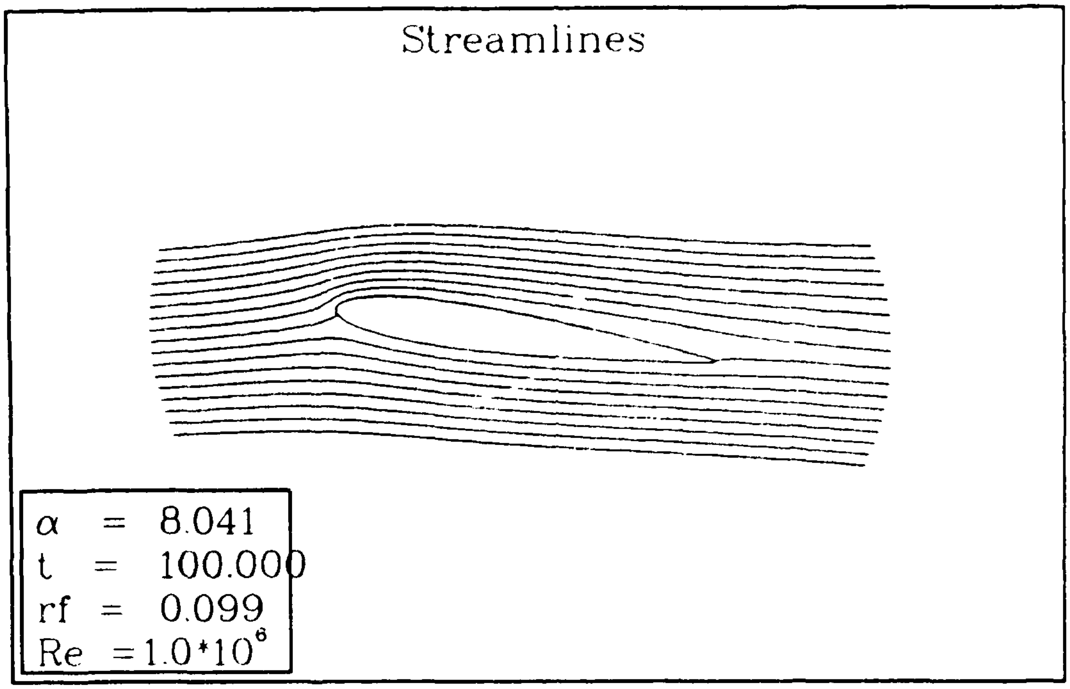

Vorticity Contours
α = 13.884
t = 88.000
rf = 0.099
Re =1.0*10^6

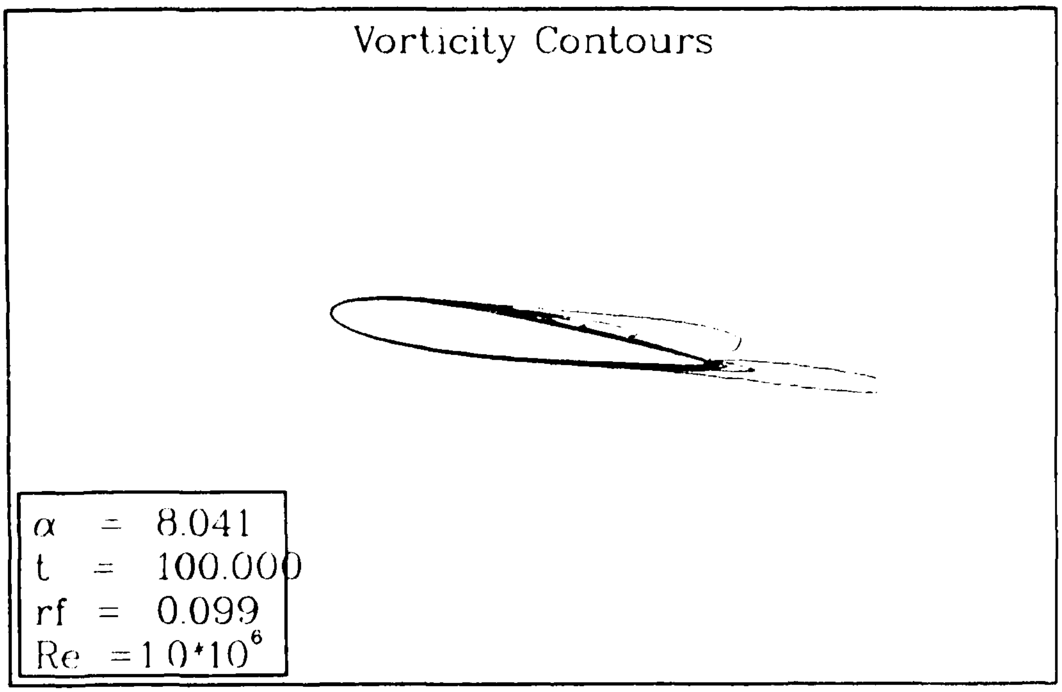

Streamlines
α = 11.749
t = 92.000
rf = 0.099
Re =1.0*10^6

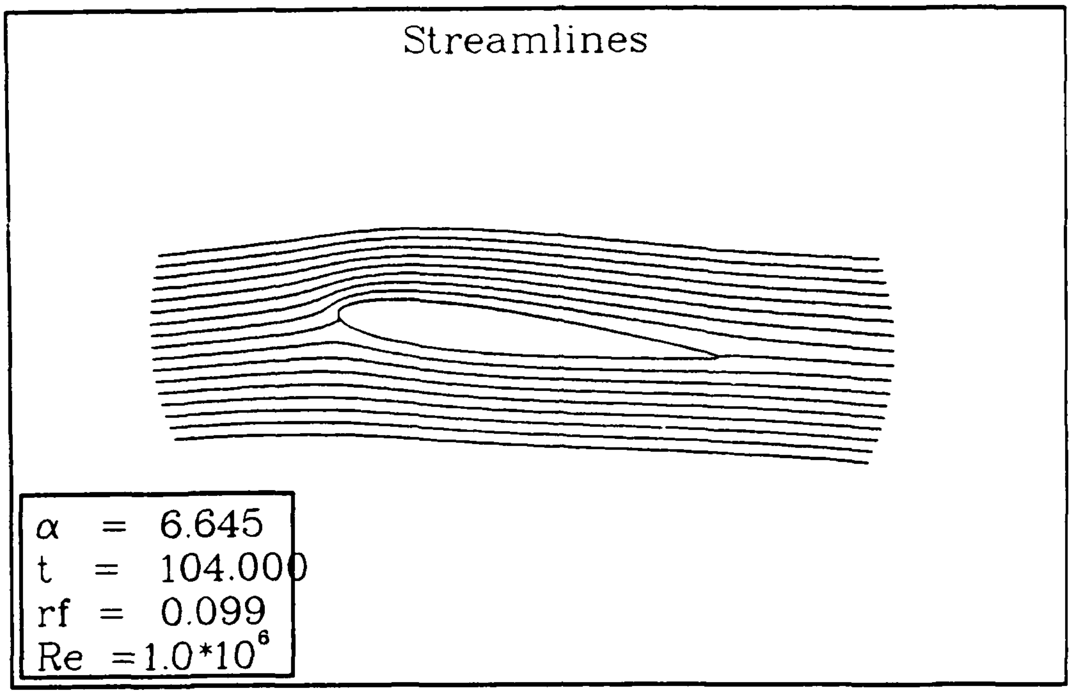

Vorticity Contours
α = 11.749
t = 92.000
rf = 0.099
Re =1.0*10^6


Streamlines
α = 9.770
t = 96.000
rf = 0.099
Re = 1.0*10^6

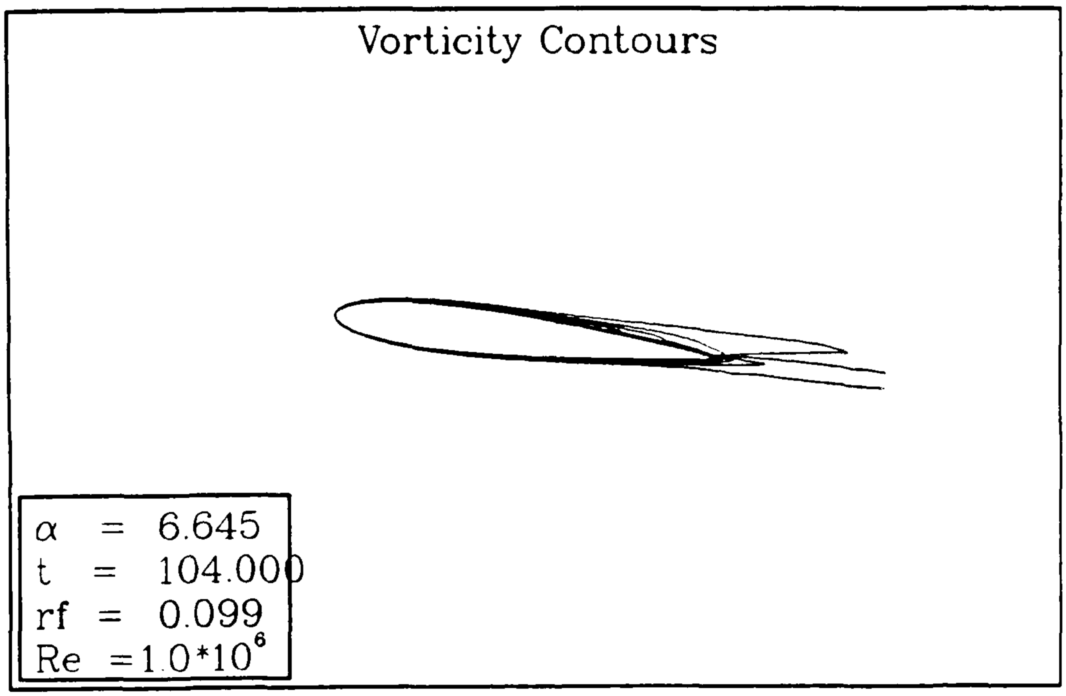

Vorticity Contours
α = 9.770
t = 96.000
rf = 0.099
Re = 1.0*10^6


Streamlines
α = 8.041
t = 100.000
rf = 0.099
Re =1.0*10^6

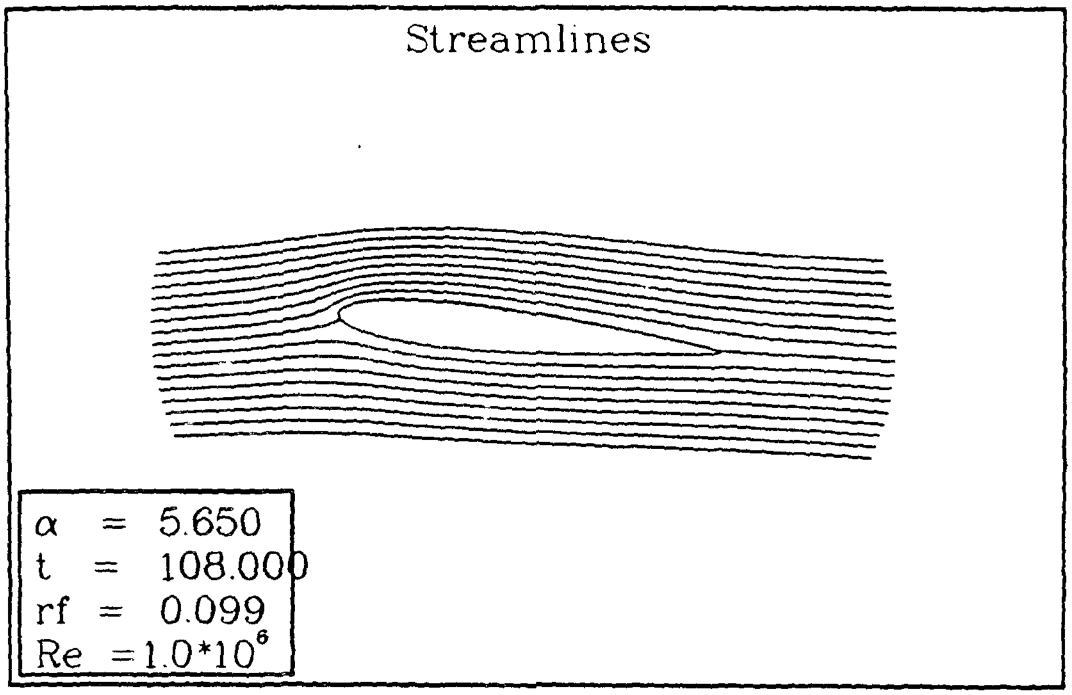

Vorticity Contours
α = 8.041
t = 100.000
rf = 0.099
Re =1.0*10^6

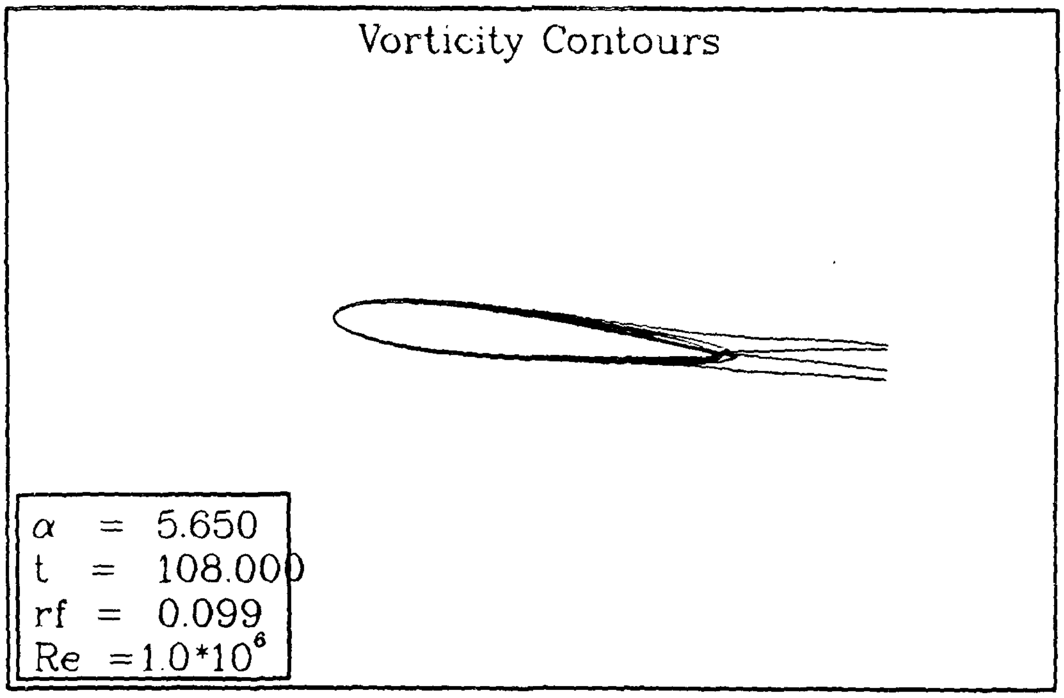


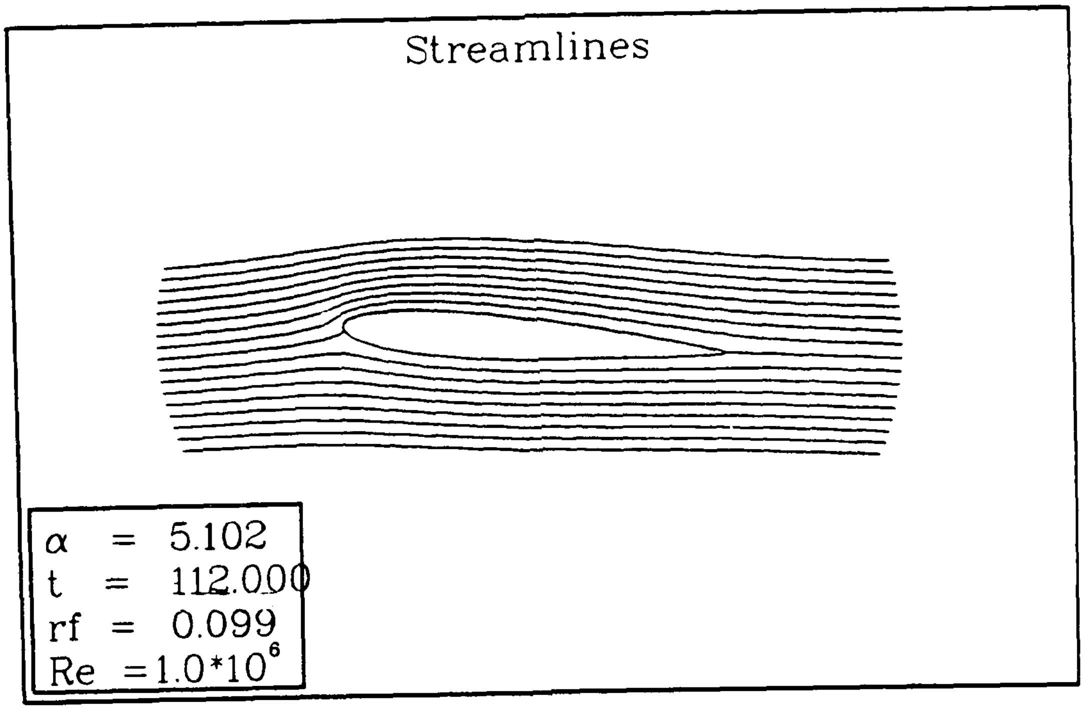
Streamlines
α = 6.645
t = 104.000
rf = 0.099
Re = 1.0*10^6

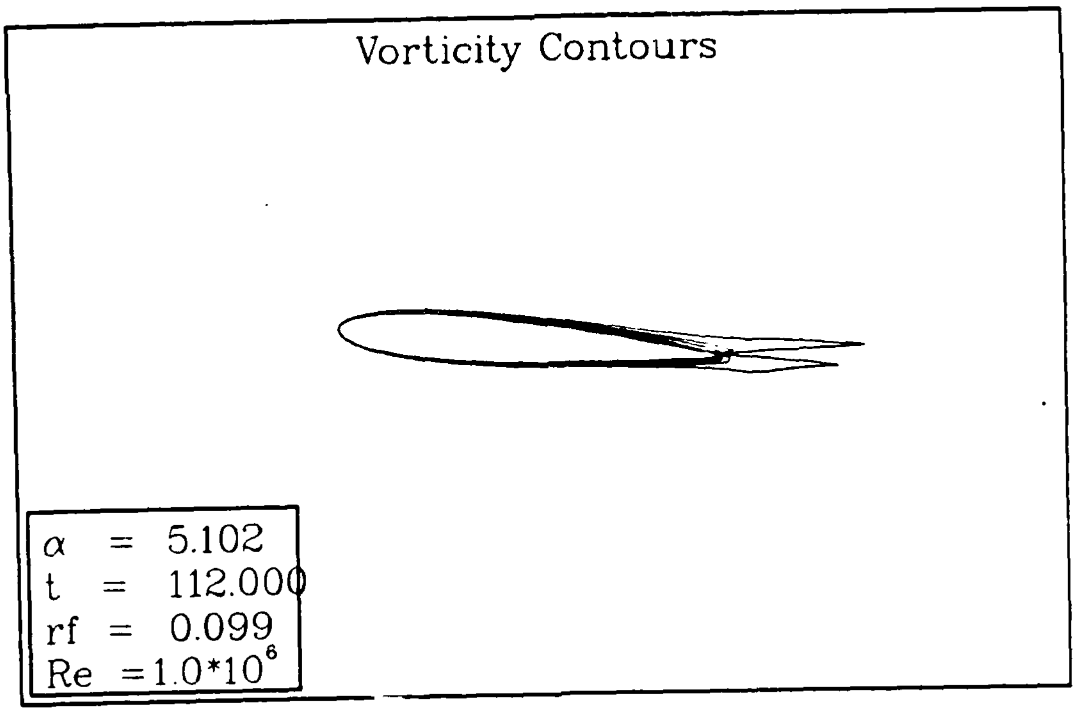


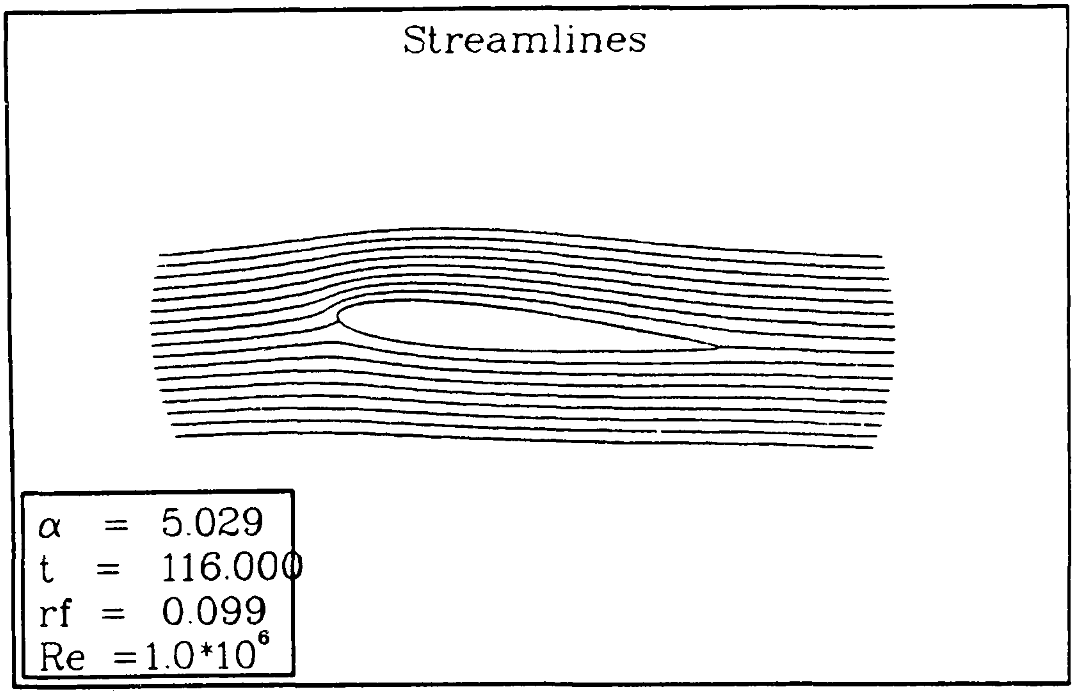
Vorticity Contours
α = 6.645
t = 104.000
rf = 0.099
Re = 1.0*10^6

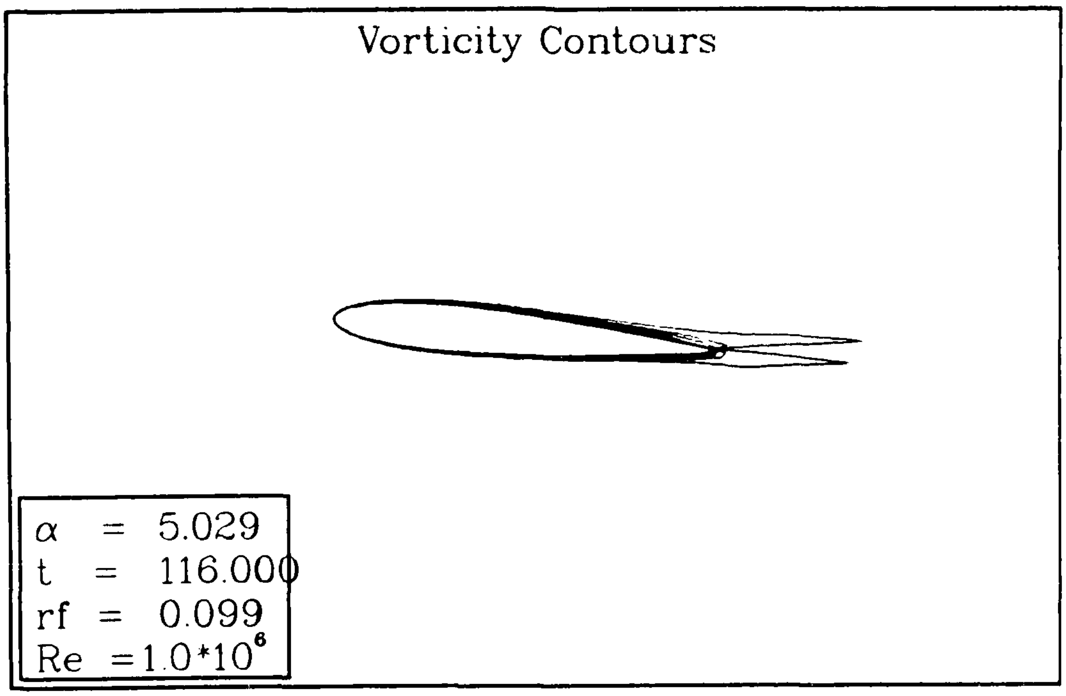

Streamlines
α = 5.650
t = 108.000
rf = 0.099
Re =1.0*10^6

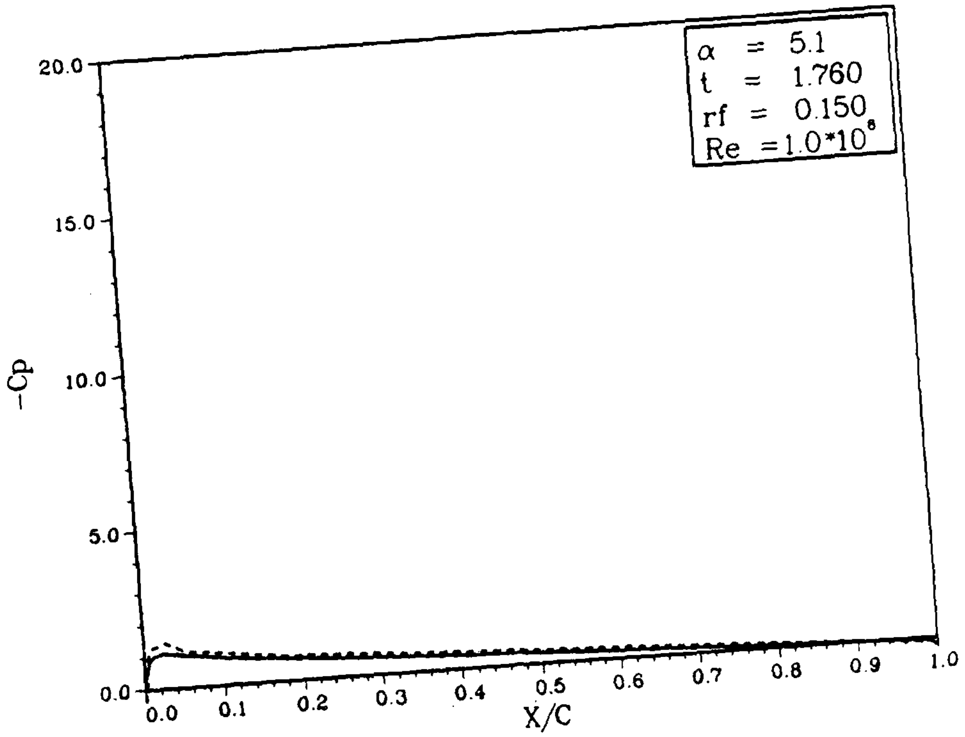

Vorticity Contours
α = 5.650
t = 108.000
rf = 0.099
Re =1.0*10^6


Streamlines
α = 5.102
t = 112.000
rf = 0.099
Re =1.0*10^6

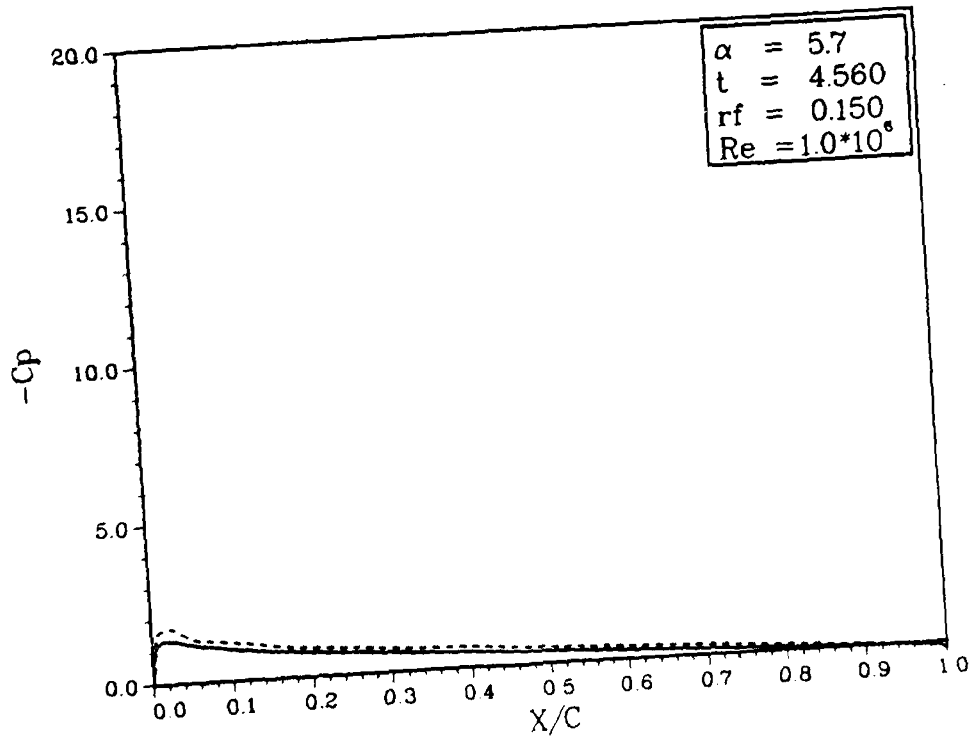

Vorticity Contours
α = 5.102
t = 112.000
rf = 0.099
Re =1.0*10^6


Streamlines
α = 5.029
t = 116.000
rf = 0.099
Re =1.0*10^6

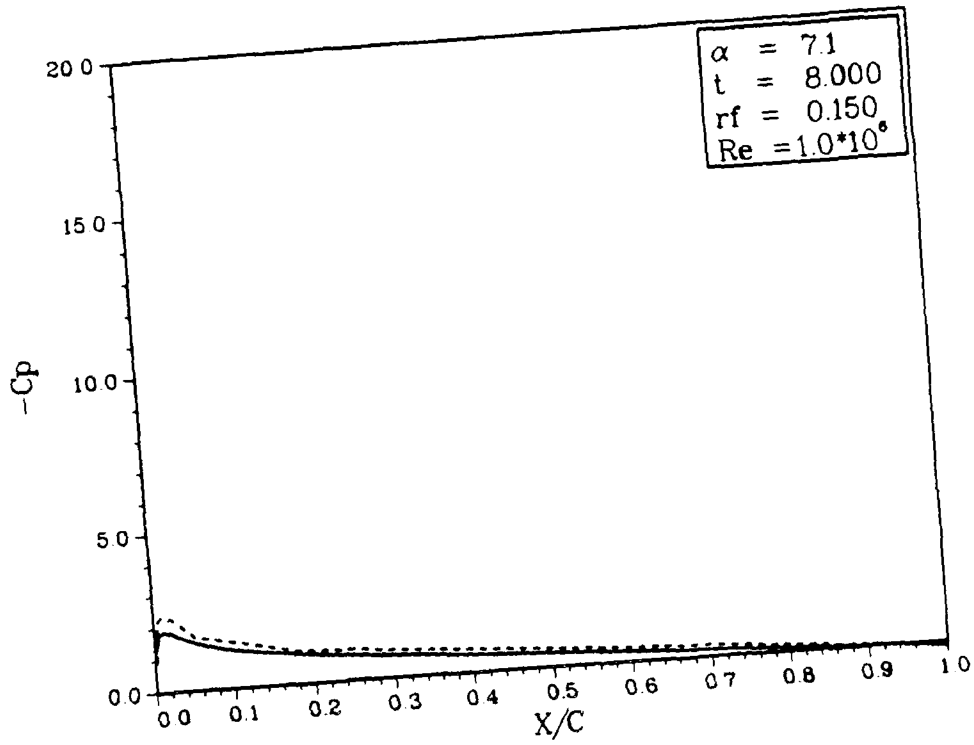

Vorticity Contours
α = 5.029
t = 116.000
rf = 0.099
Re =1.0*10^6

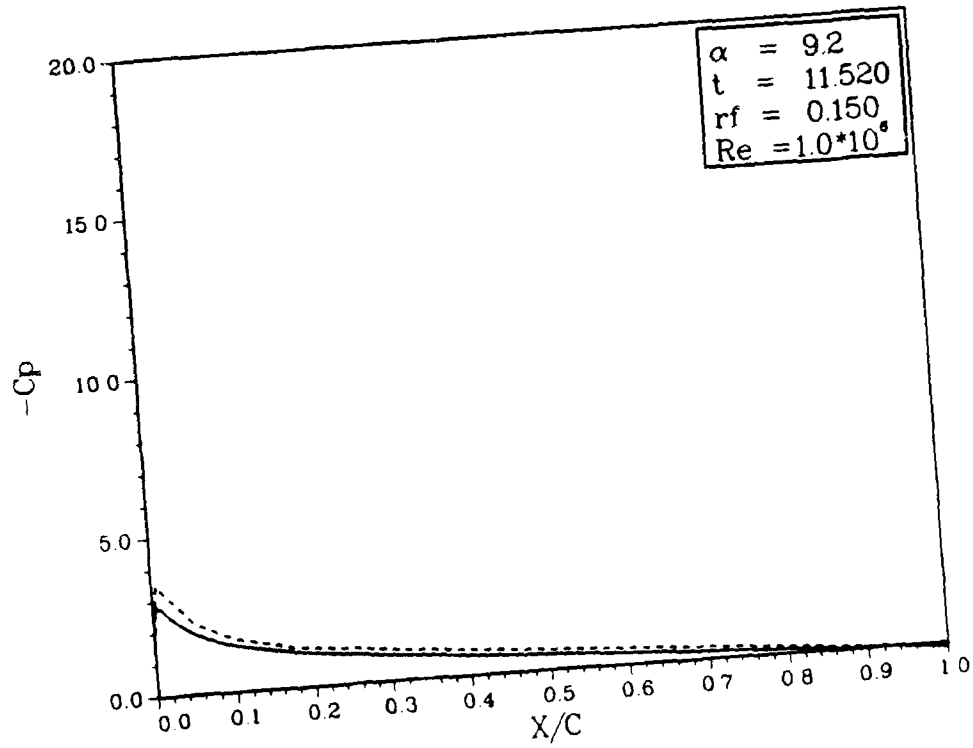


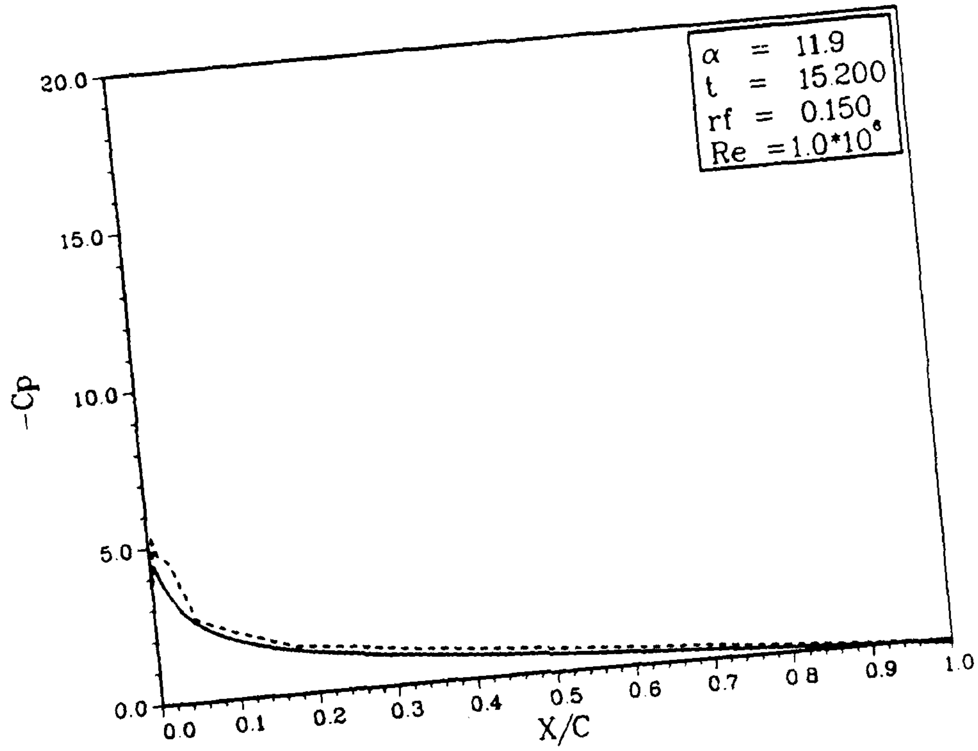
α = 5.1
t = 1.760
rf = 0.150
Re =1.0*10^6
-Cp
X/C

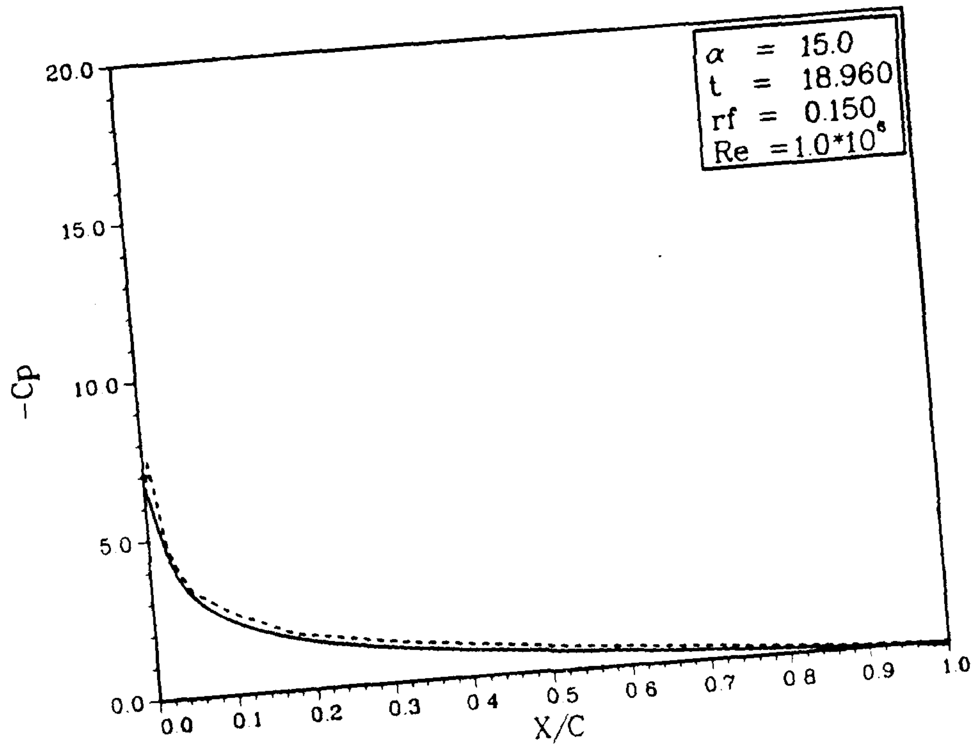


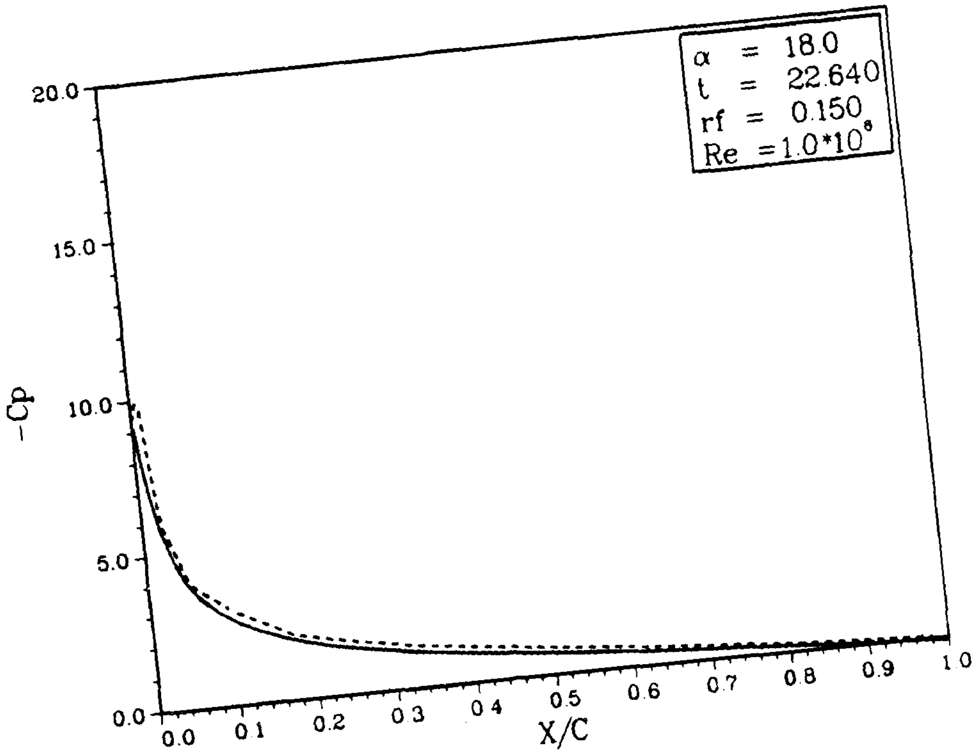
α = 5.7
t = 4.560
rf = 0.150
Re =1.0*10^6
-Cp
X/C

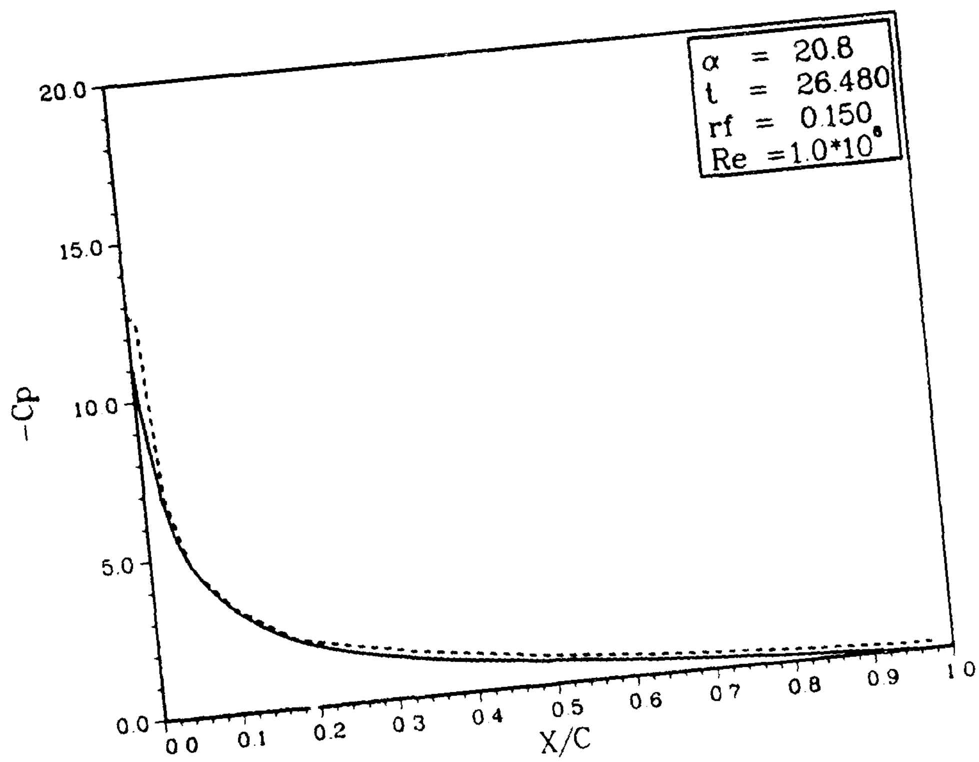

α = 7.1
t = 8.000
rf = 0.150
Re =1.0*10^6
-Cp
X/C

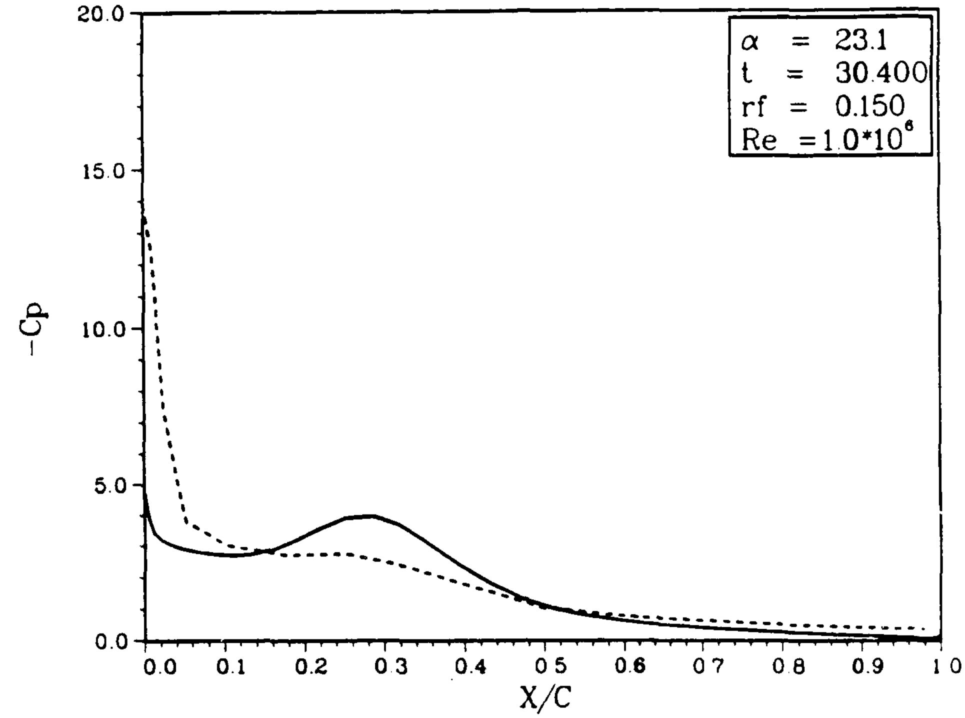

α = 9.2
t = 11.520
rf = 0.150
Re =1.0*10^6
-Cp
X/C

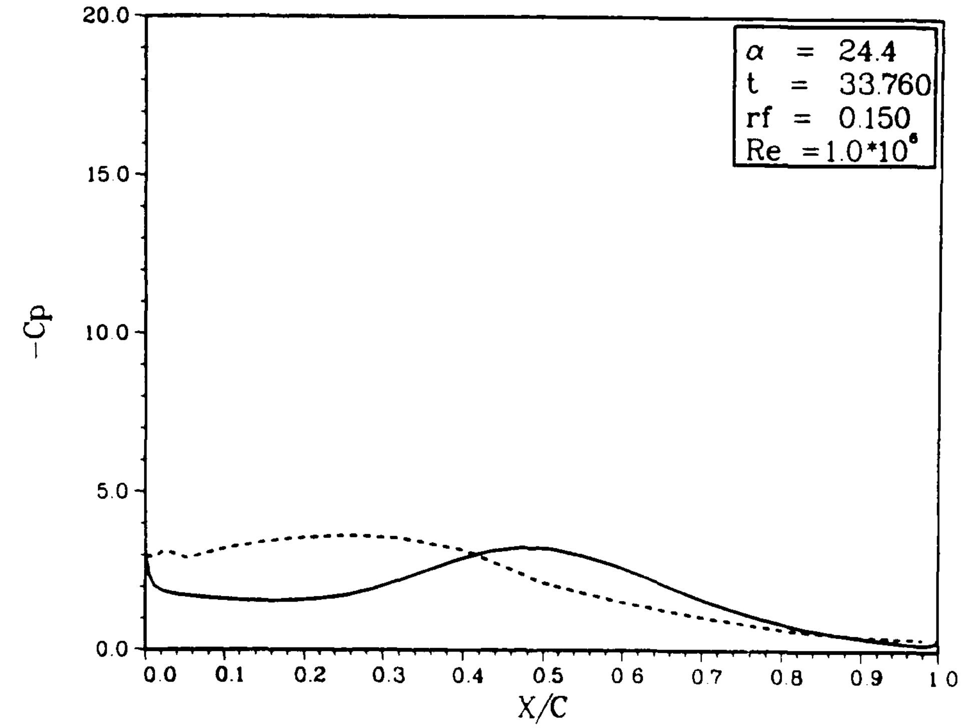

α = 11.9
t = 15.200
rf = 0.150
Re = 1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

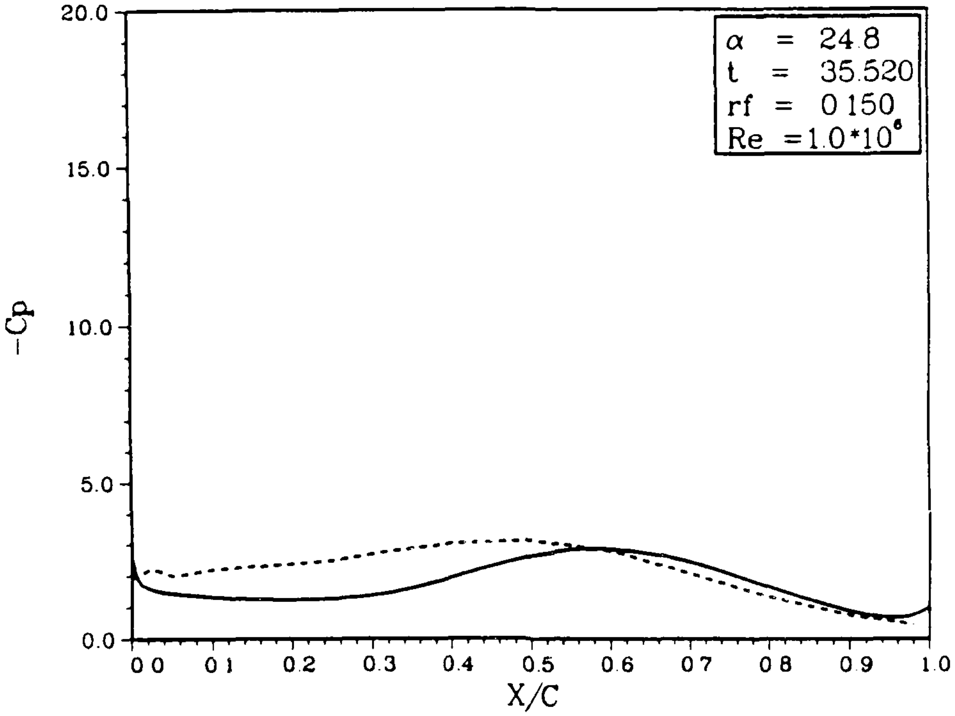

α = 15.0
t = 18.960
rf = 0.150
Re = 1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

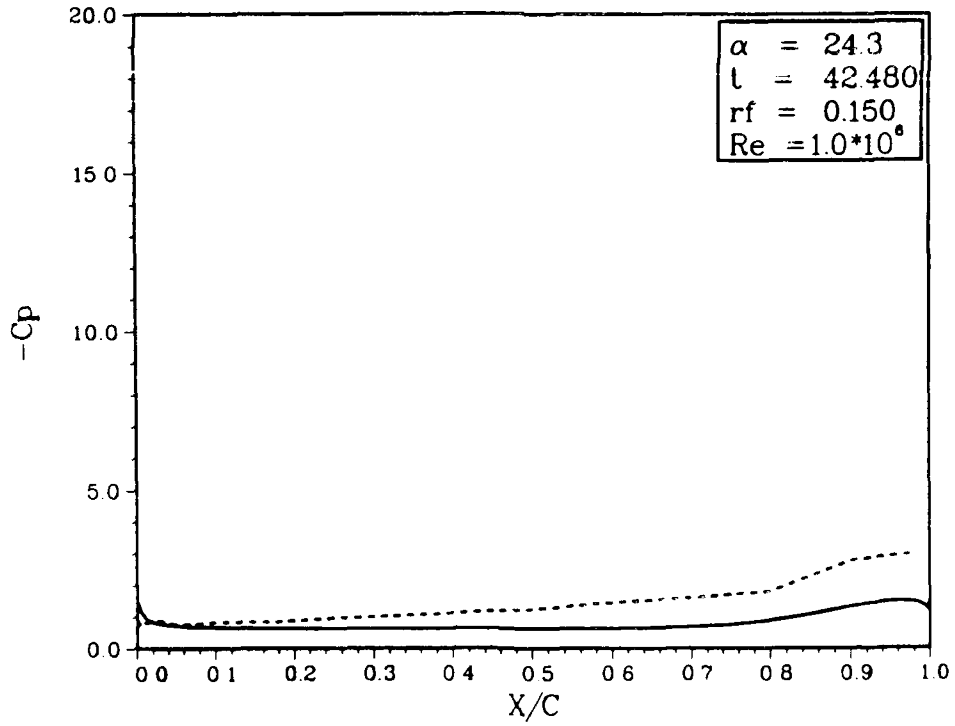

α = 18.0
t = 22.640
rf = 0.150
Re =1.0*10^6
-Cp
X/C

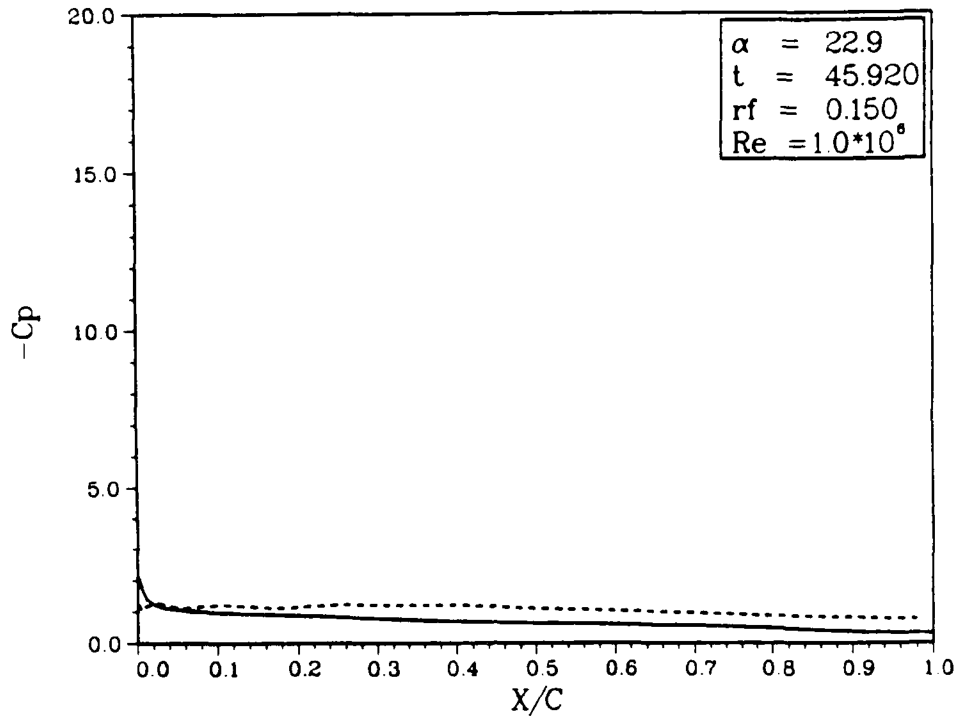

α = 20.8
t = 26.480
rf = 0.150
Re =1.0*10^6
-Cp
X/C

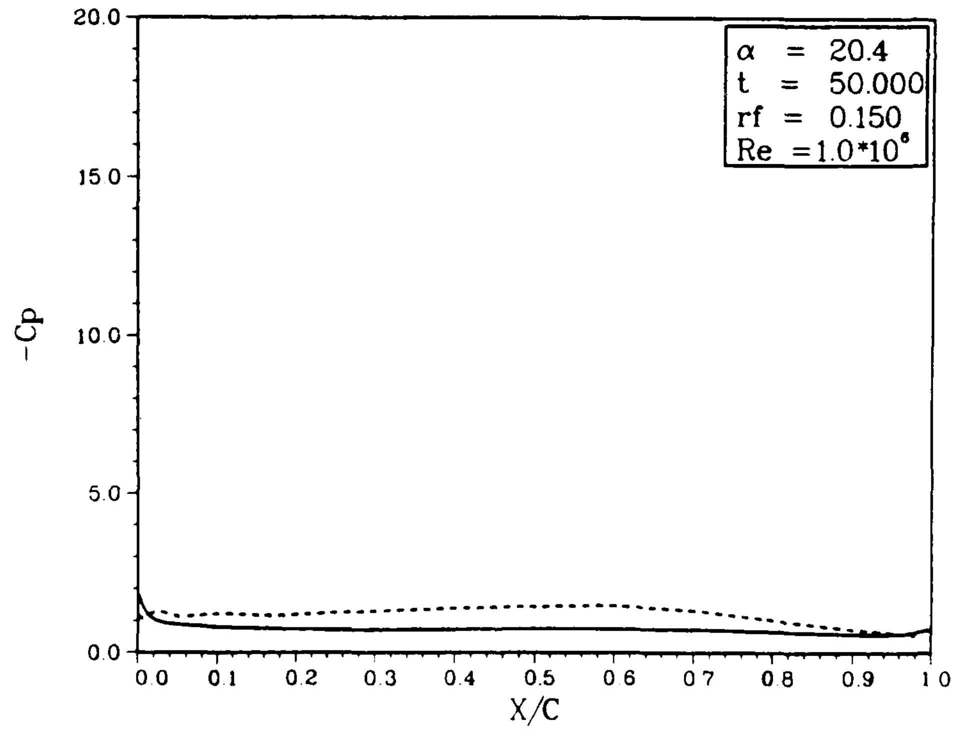

20.0
15.0
10.0
5.0
0.0
-Cp
α = 23.1
t = 30.400
rf = 0.150
Re =1.0*10⁶
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

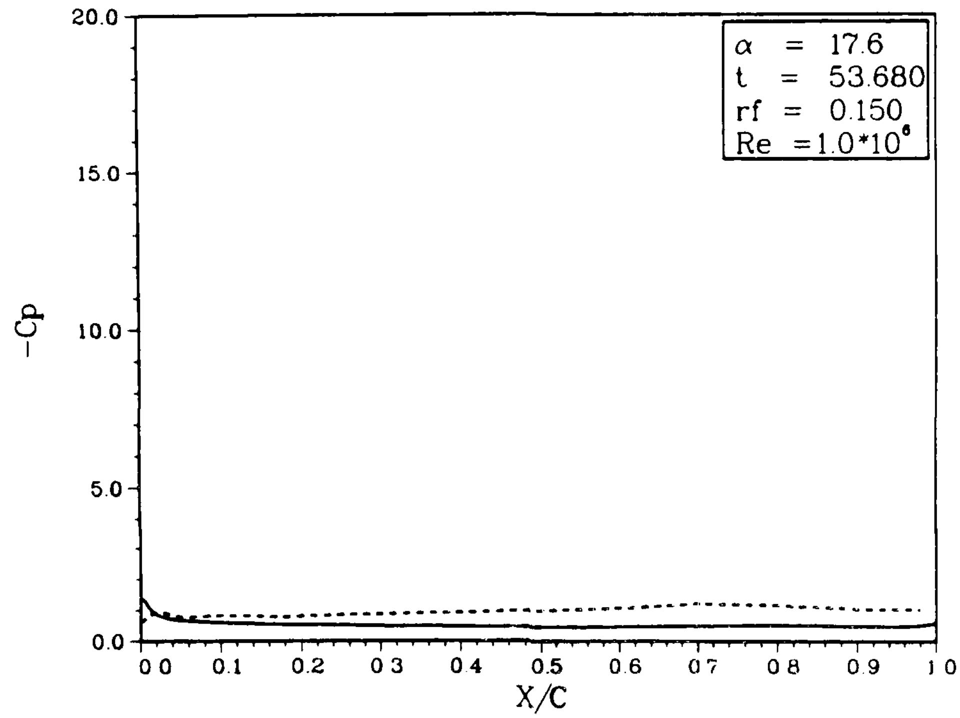

20.0
15.0
10.0
5.0
0.0
-Cp
α = 24.4
t = 33.760
rf = 0.150
Re =1.0*10⁶
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C


α = 24.8
t = 35.520
rf = 0.150
Re = 1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

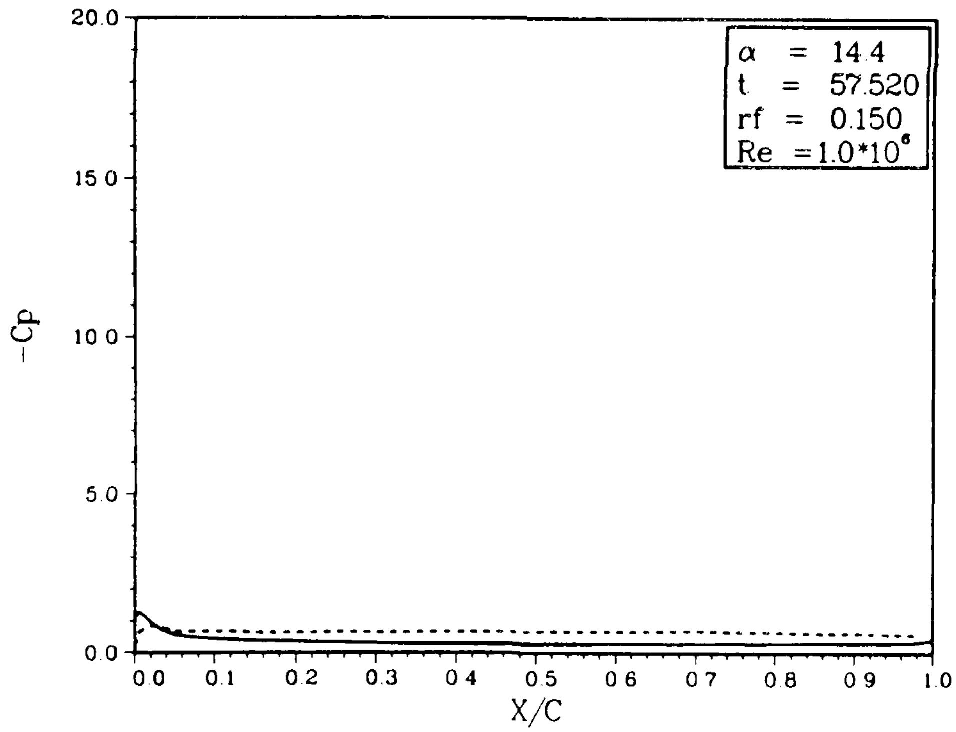

α = 24.3
t = 42.480
rf = 0.150
Re = 1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0


α = 22.9
t = 45.920
rf = 0.150
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

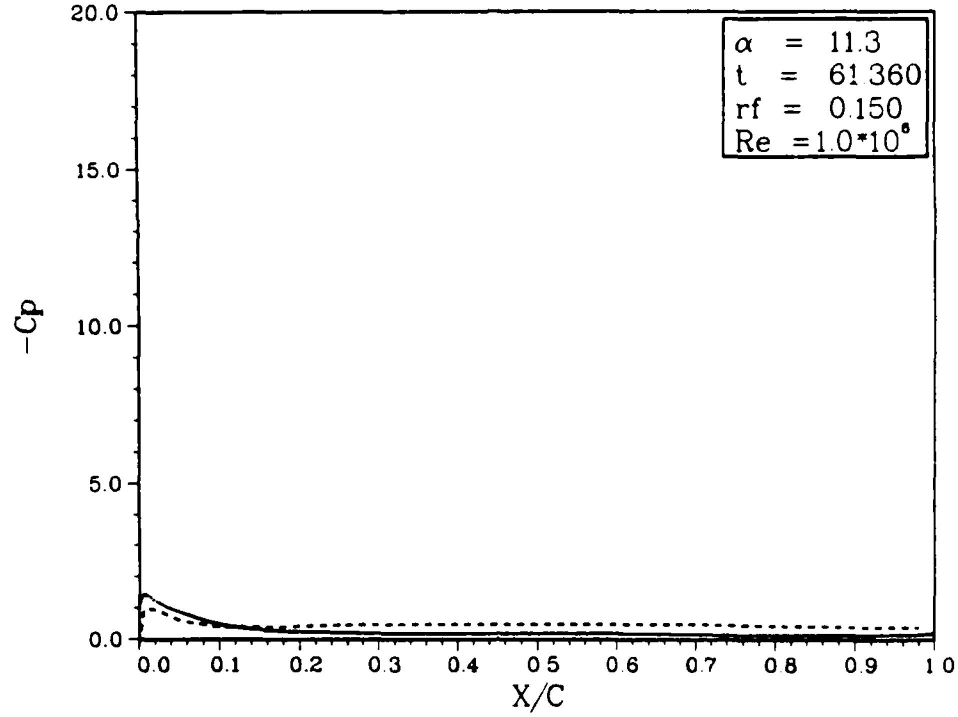

α = 20.4
t = 50.000
rf = 0.150
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C



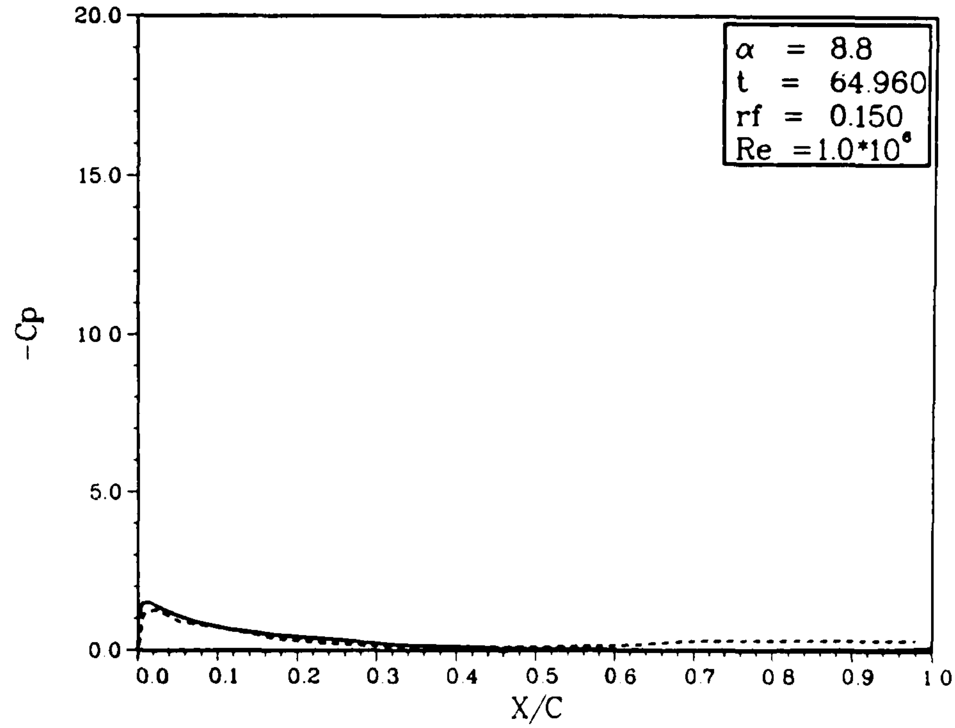
α = 17.6
t = 53.680
rf = 0.150
Re = 1.0*10^6
-Cp
X/C



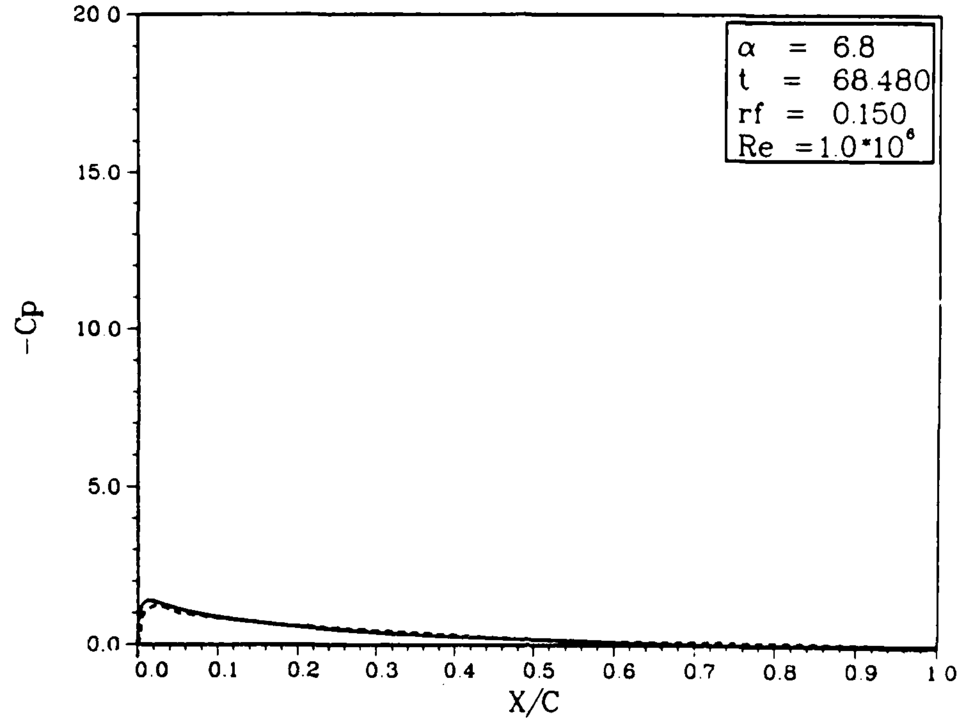
α = 14.4
t = 57.520
rf = 0.150
Re = 1.0*10^6
-Cp
X/C

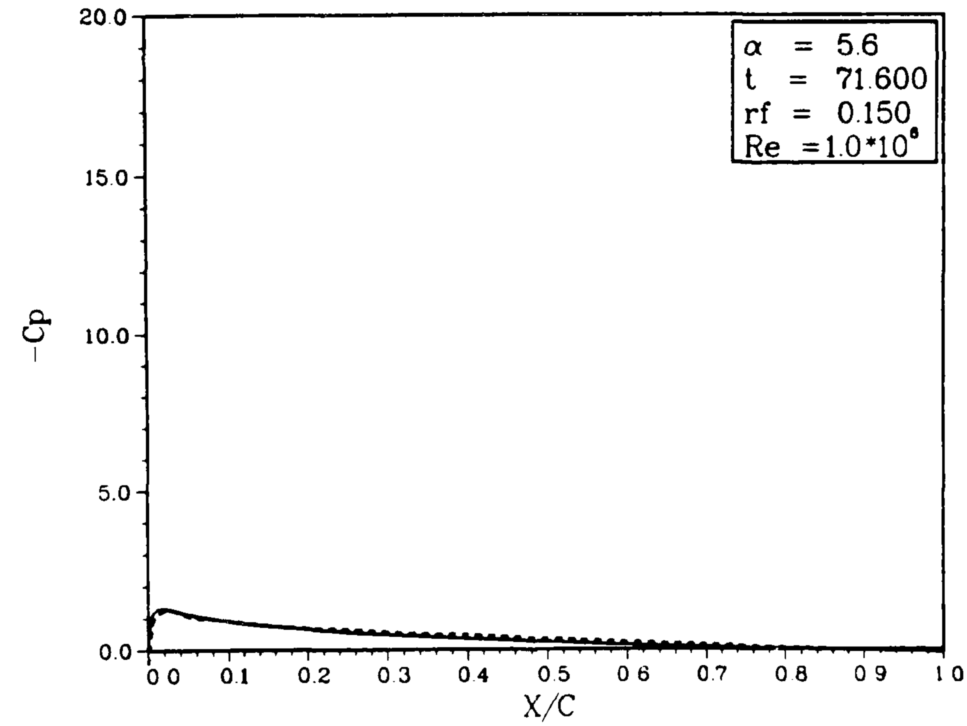


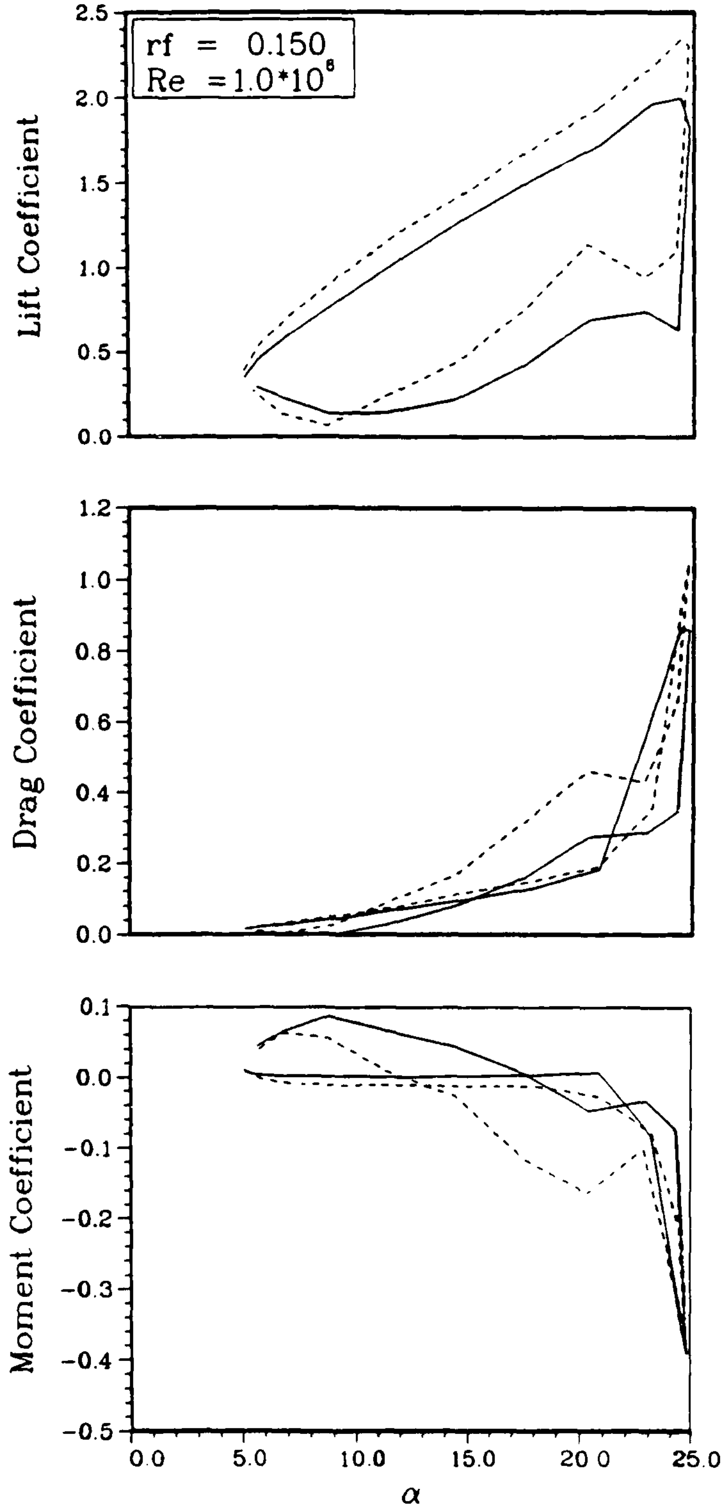
α = 11.3
t = 61.360
rf = 0.150
Re =1.0*10^6
-Cp
X/C



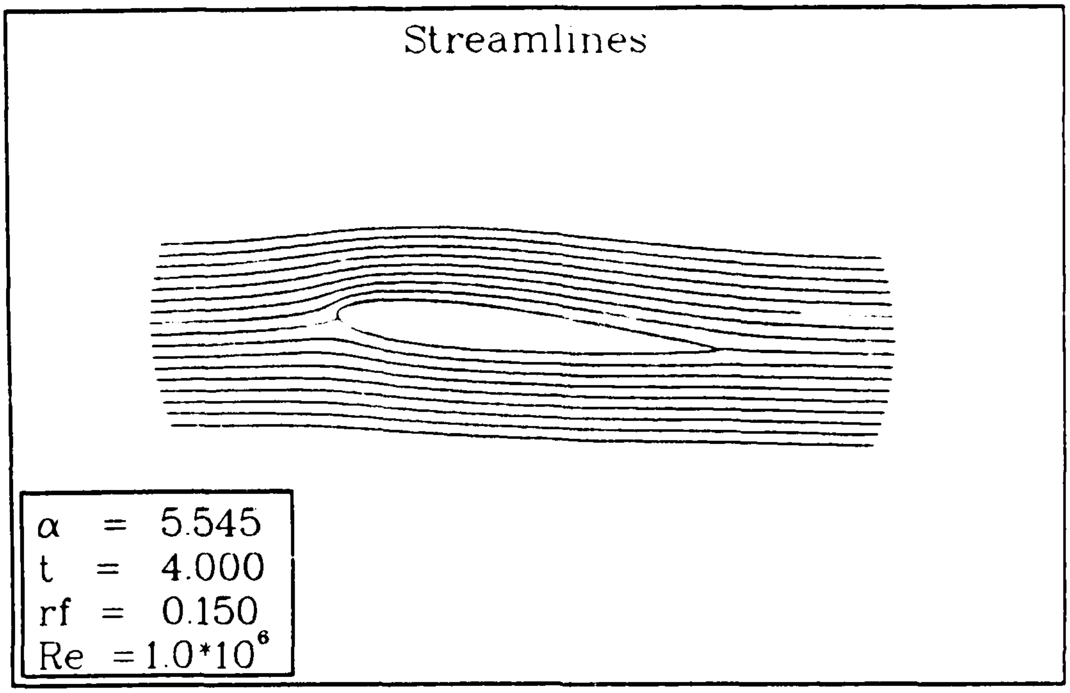
α = 8.8
t = 64.960
rf = 0.150
Re =1.0*10^6
-Cp
X/C

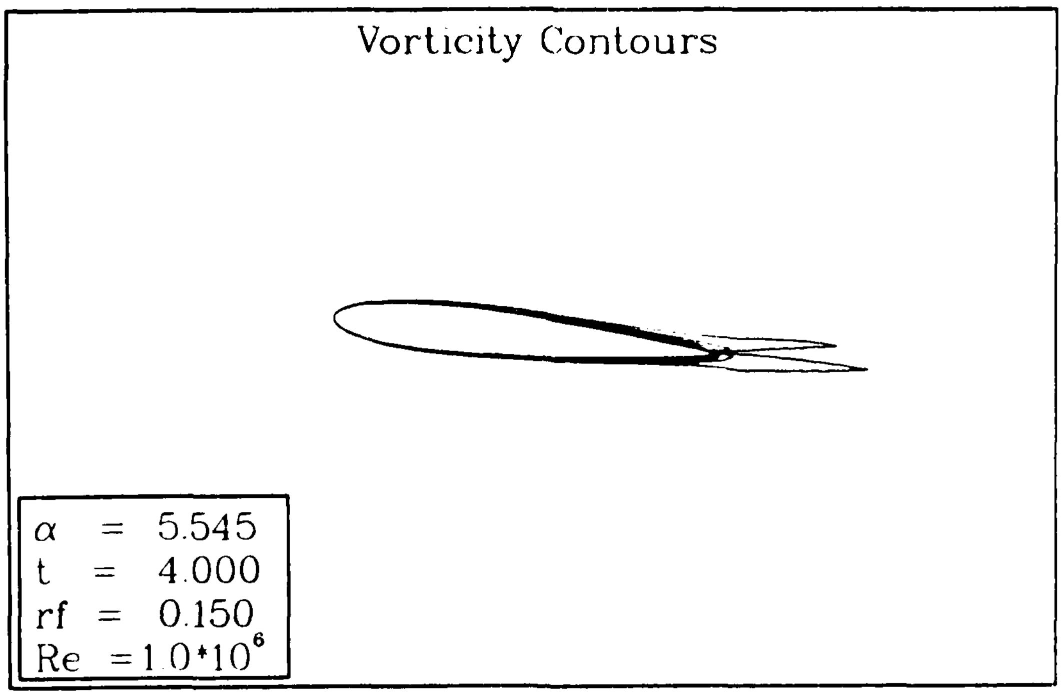

α = 6.8
t = 68.480
rf = 0.150
Re =1.0*10^6
-Cp
X/C

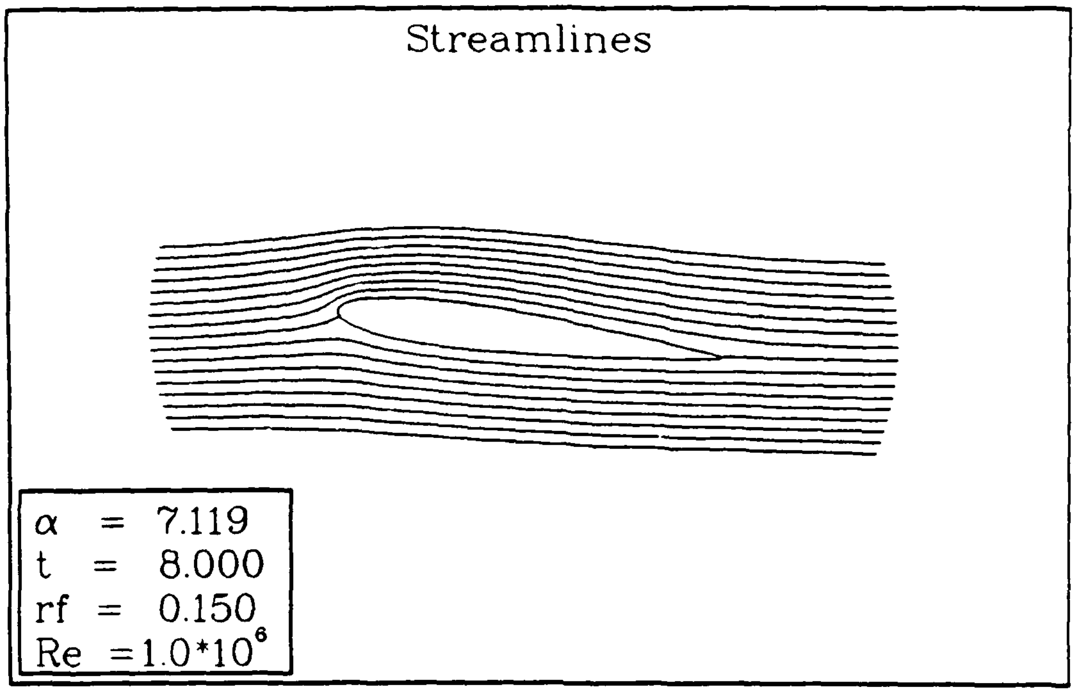

α = 5.6
t = 71.600
rf = 0.150
Re =1.0*10^6
-Cp
X/C

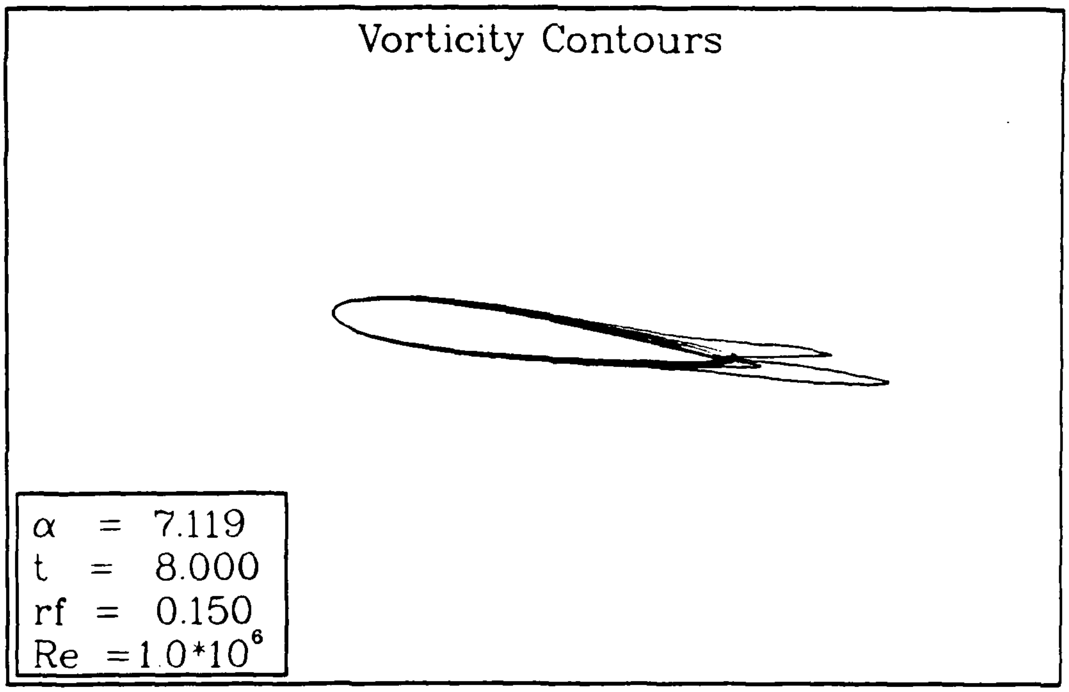


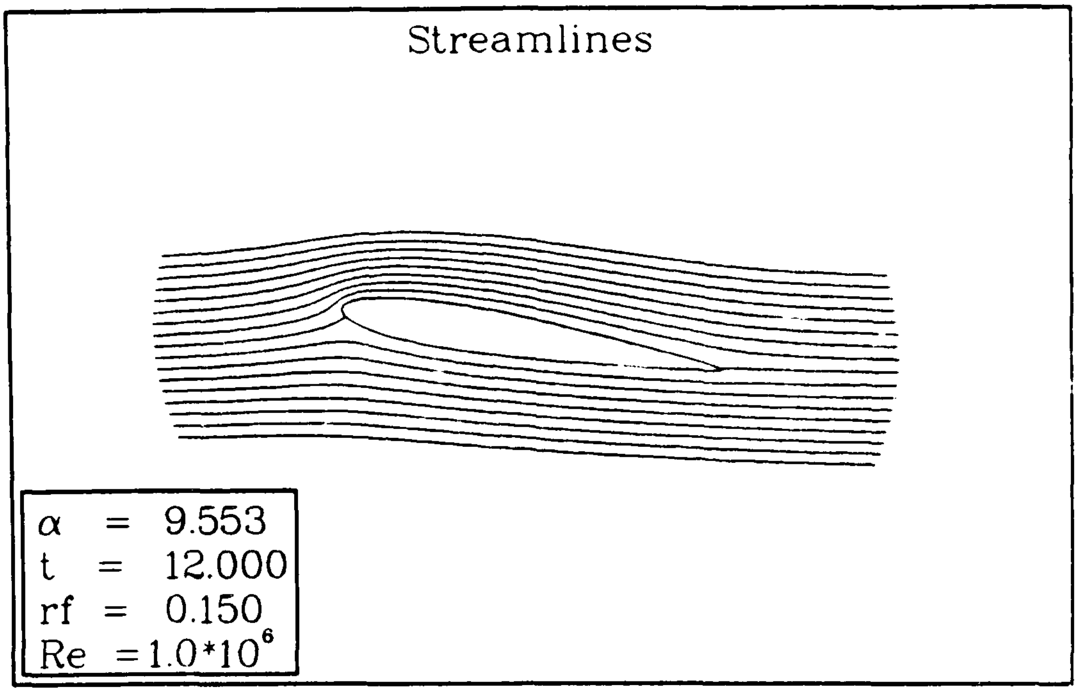
rf = 0.150
Re =1.0*10^6
Lift Coefficient
2.5
2.0
1.5
1.0
0.5
0.0
Drag Coefficient
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Moment Coefficient
0.1
0.0
-0.1
-0.2
-0.3
-0.4
-0.5
0.0
5.0
10.0
15.0
20.0
25.0
α

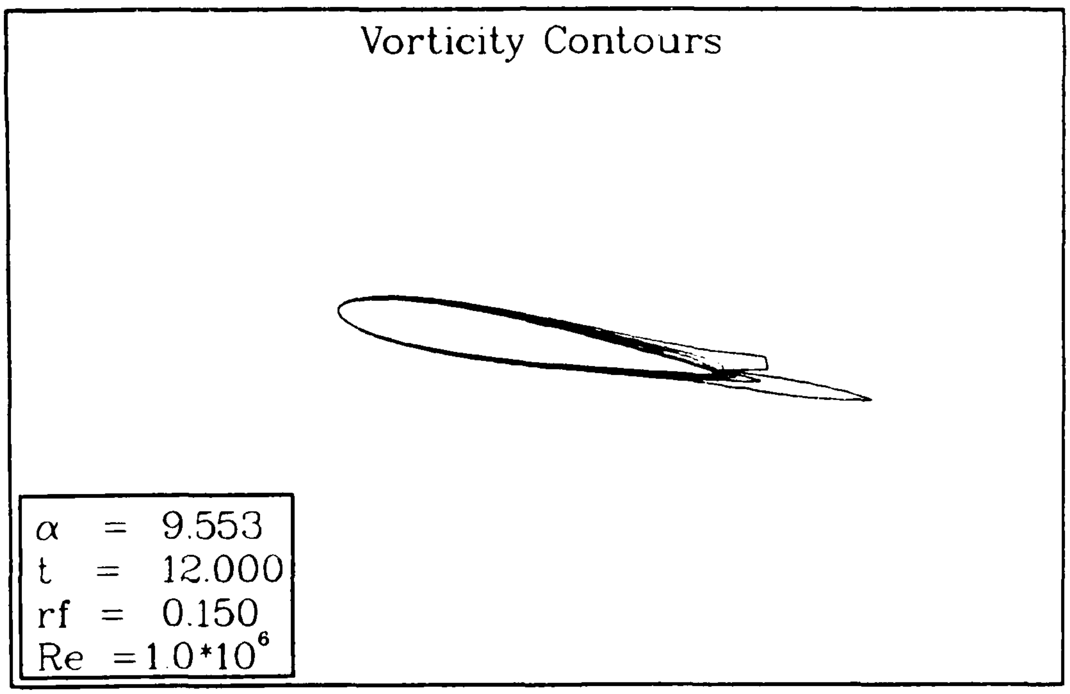

Streamlines
α = 5.545
t = 4.000
rf = 0.150
Re = 1.0*10^6

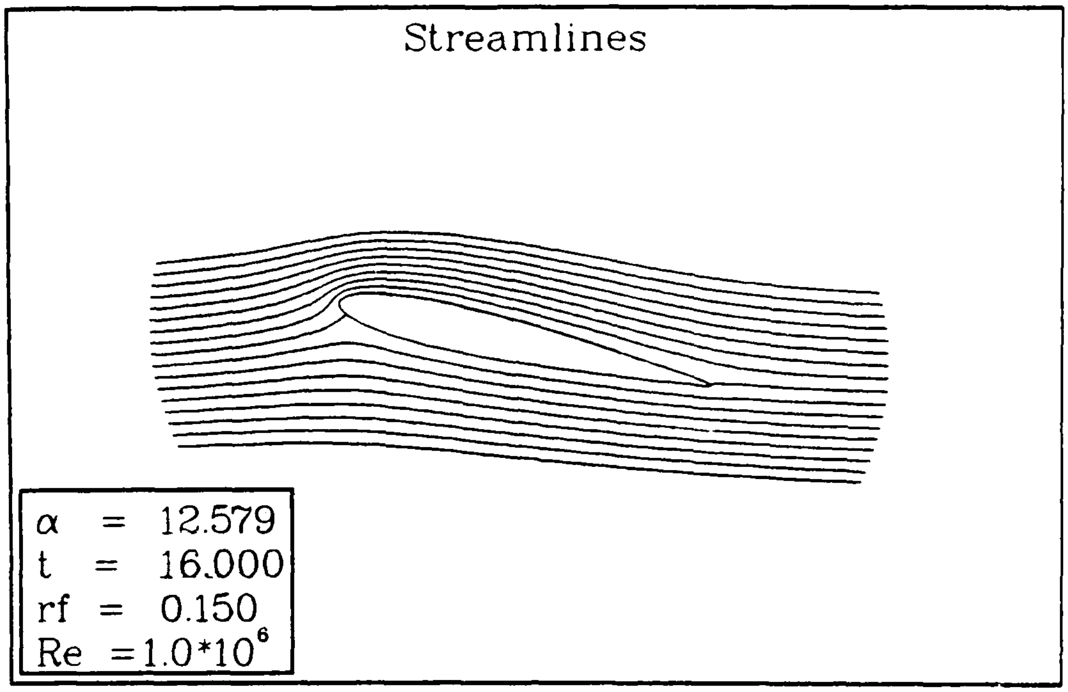

Vorticity Contours
α = 5.545
t = 4.000
rf = 0.150
Re = 1.0*10^6

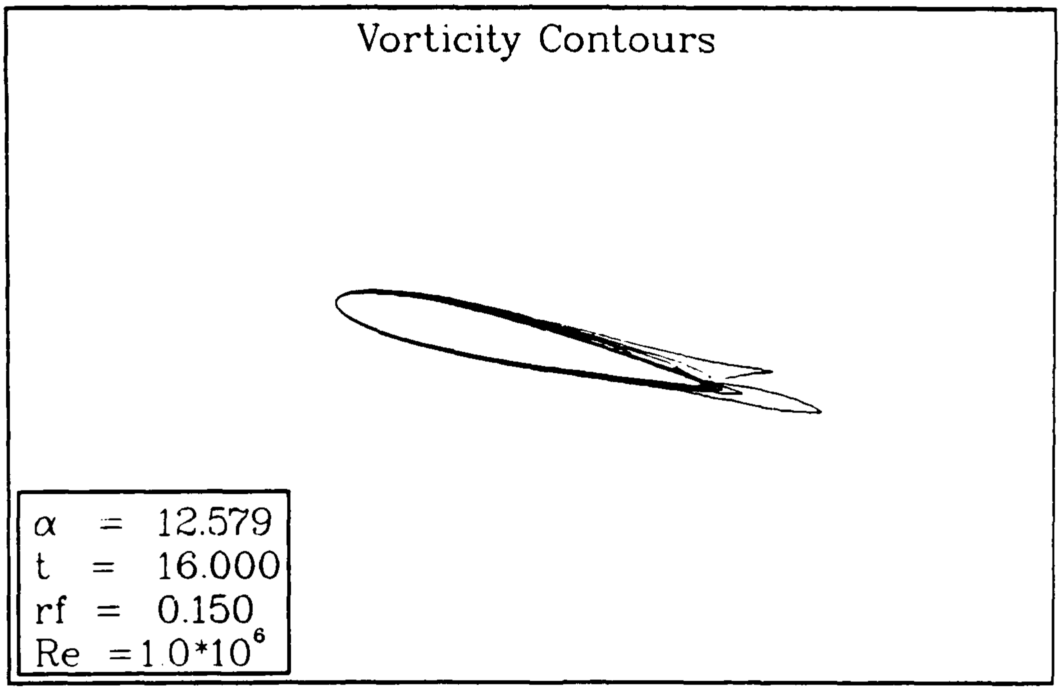

Streamlines
α = 7.119
t = 8.000
rf = 0.150
Re =1.0*10^6

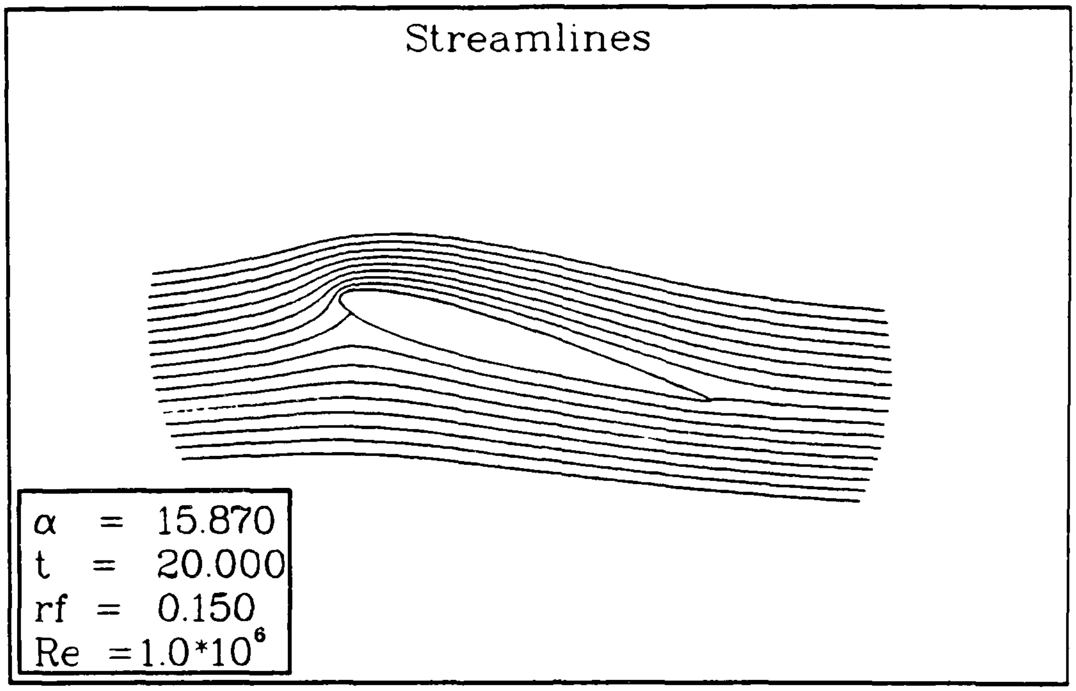

Vorticity Contours
α = 7.119
t = 8.000
rf = 0.150
Re =1.0*10^6

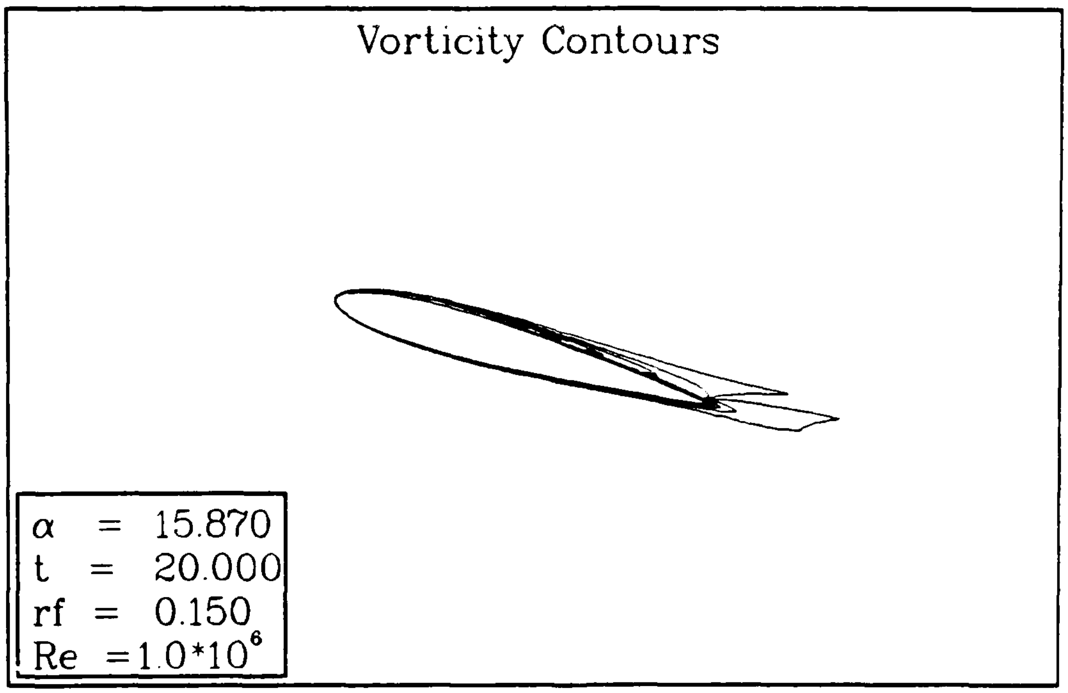

Streamlines
α = 9.553
t = 12.000
rf = 0.150
Re =1.0*10⁶

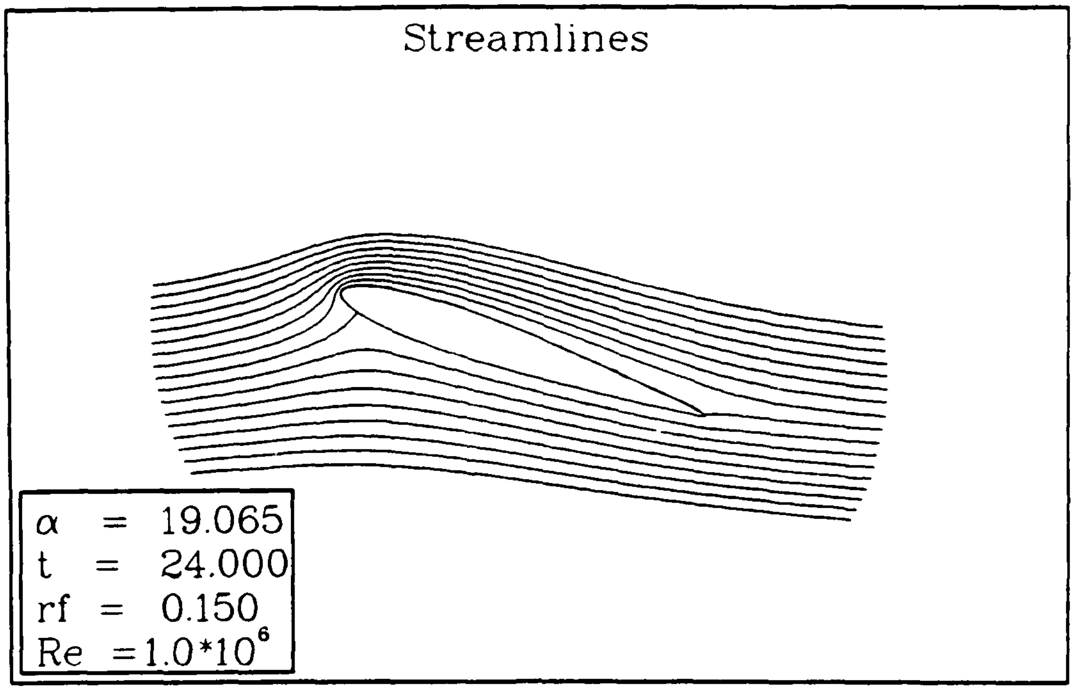

Vorticity Contours
α = 9.553
t = 12.000
rf = 0.150
Re =1.0*10⁶

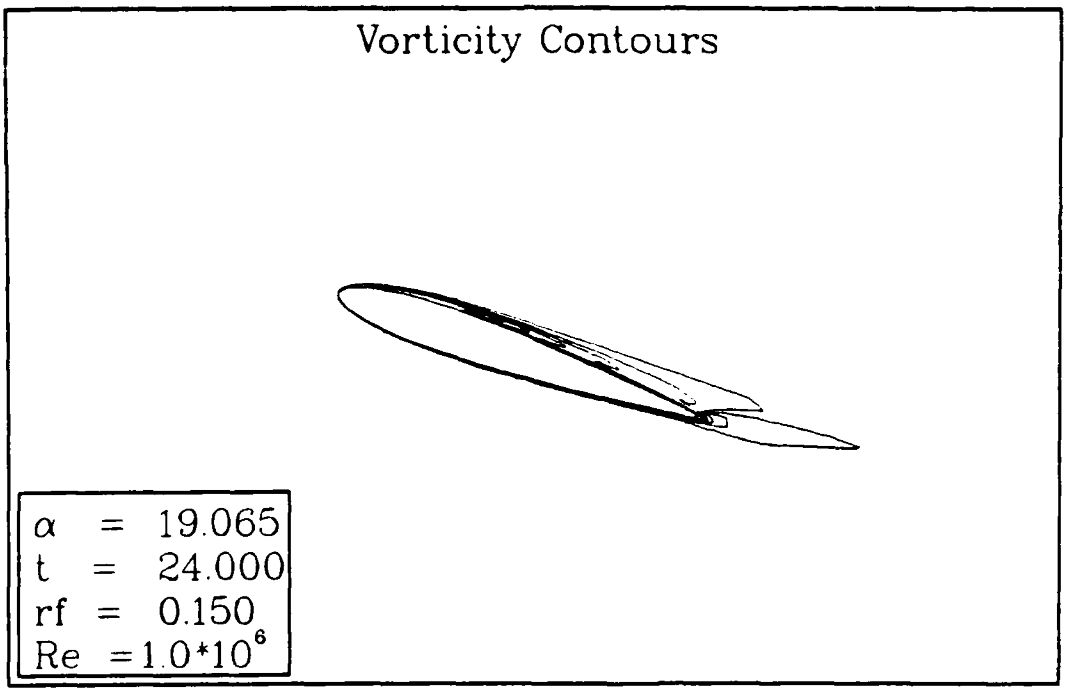

Streamlines
α = 12.579
t = 16.000
rf = 0.150
Re =1.0*10^6

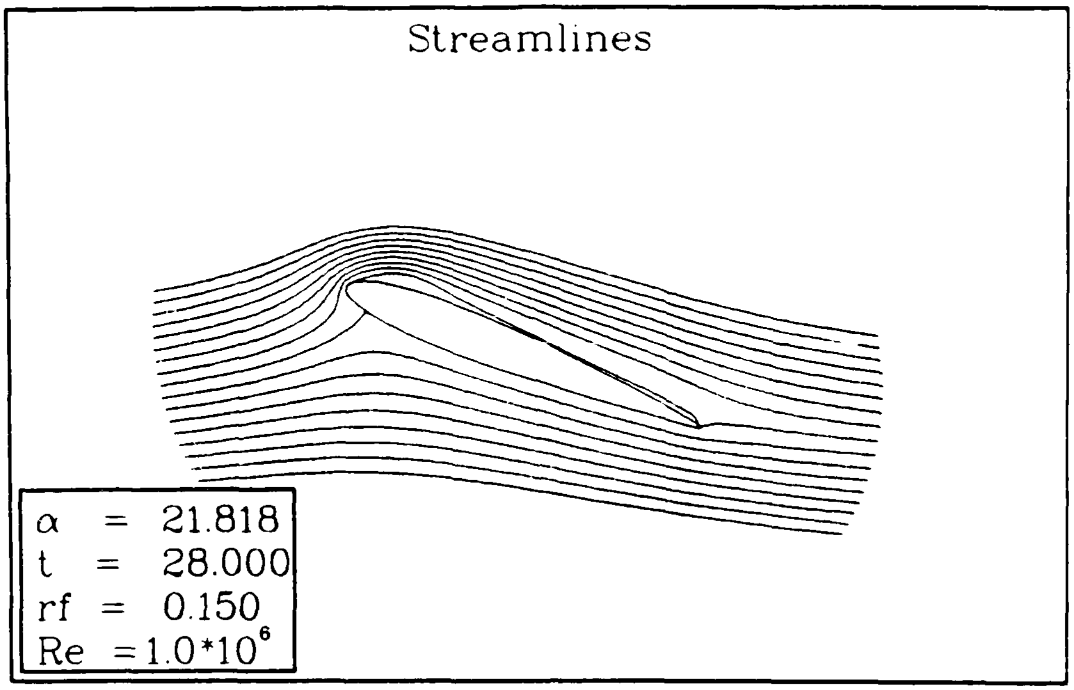

Vorticity Contours
α = 12.579
t = 16.000
rf = 0.150
Re =1.0*10^6

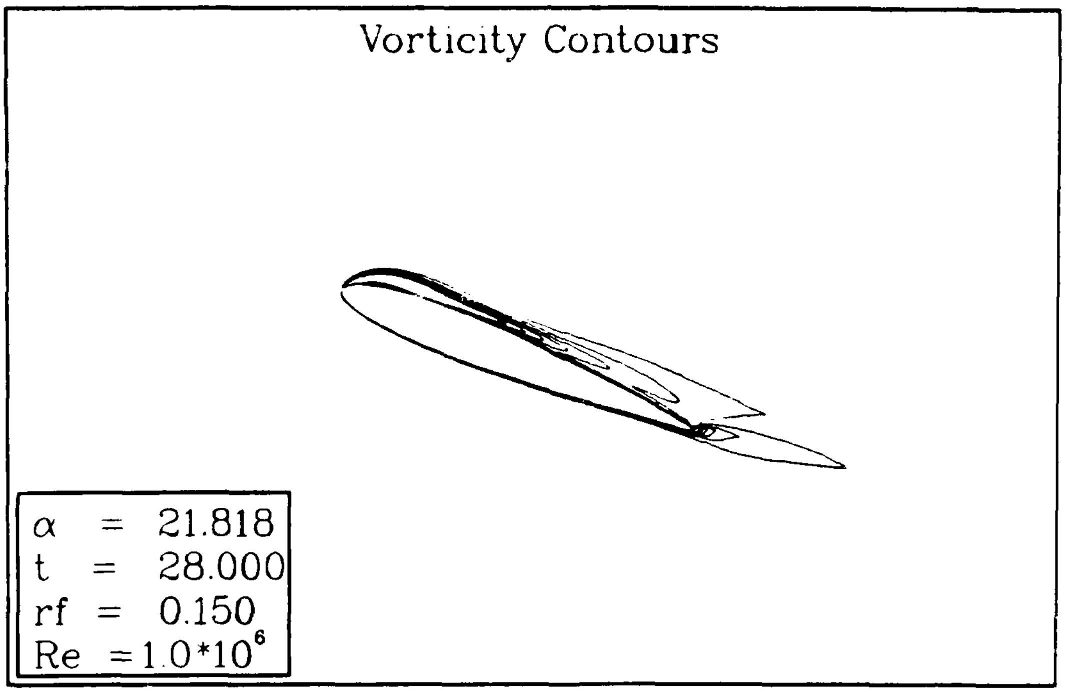

Streamlines
α = 15.870
t = 20.000
rf = 0.150
Re =1.0*10^6

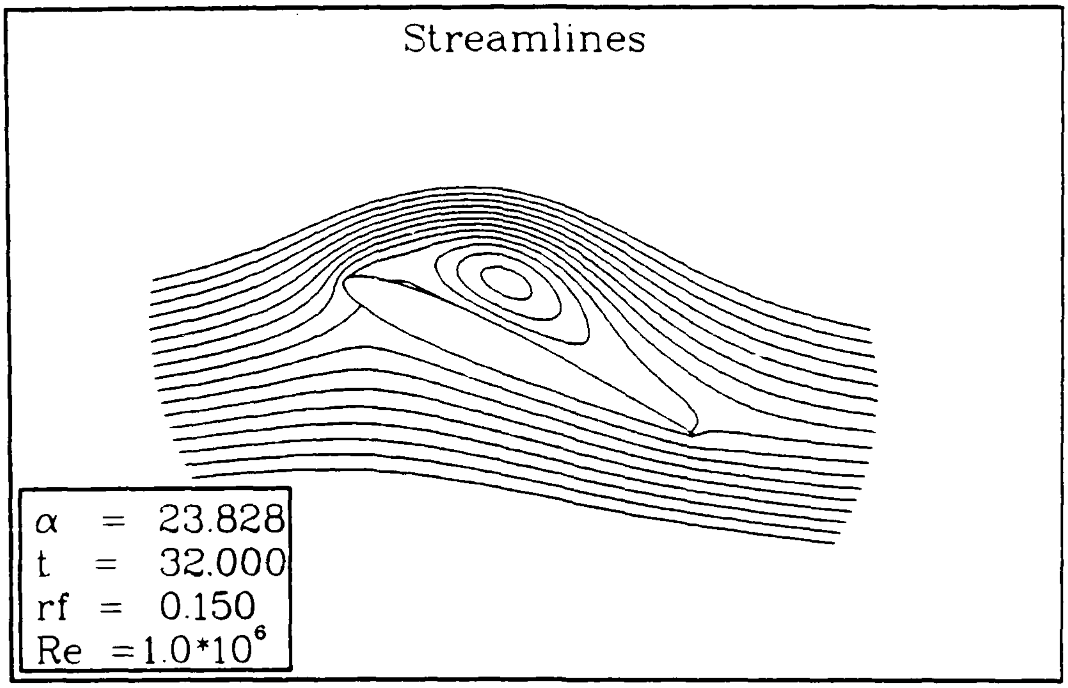

Vorticity Contours
α = 15.870
t = 20.000
rf = 0.150
Re =1.0*10^6

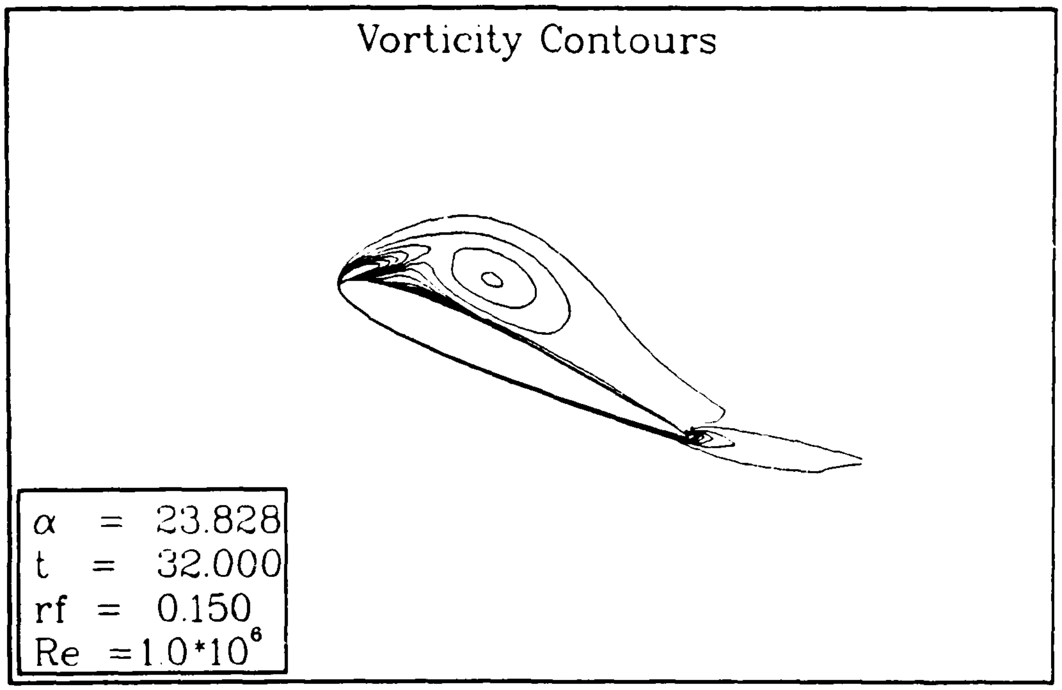

Streamlines
α = 19.065
t = 24.000
rf = 0.150
Re =1.0*10^6

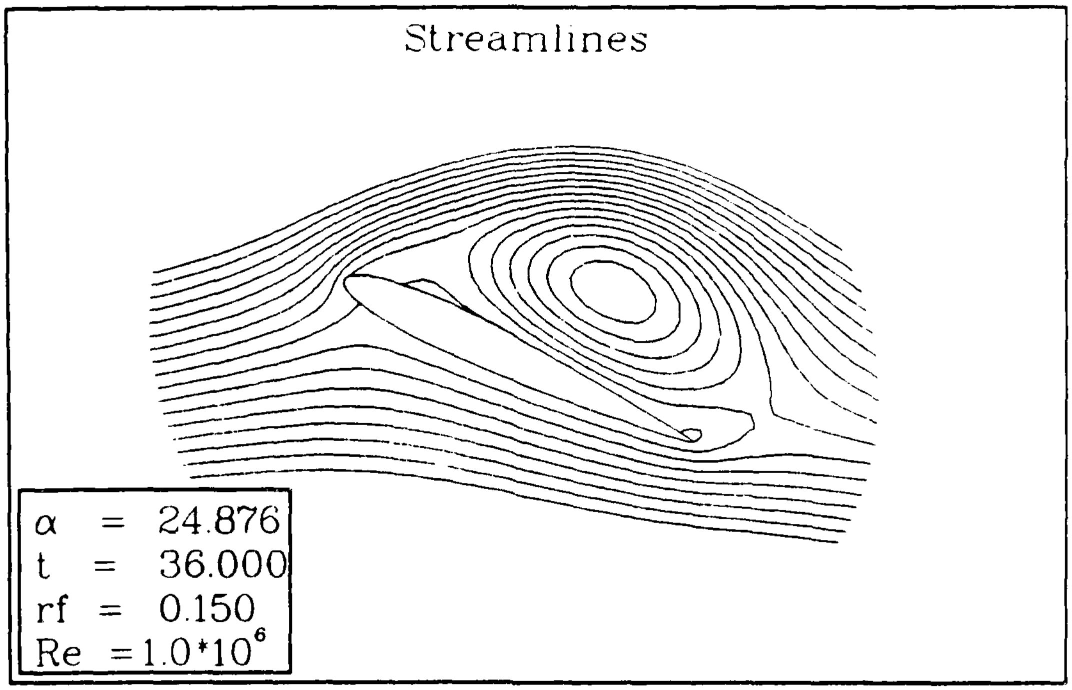

Vorticity Contours
α = 19.065
t = 24.000
rf = 0.150
Re =1.0*10^6


Streamlines
α = 21.818
t = 28.000
rf = 0.150
Re = 1.0*10^6

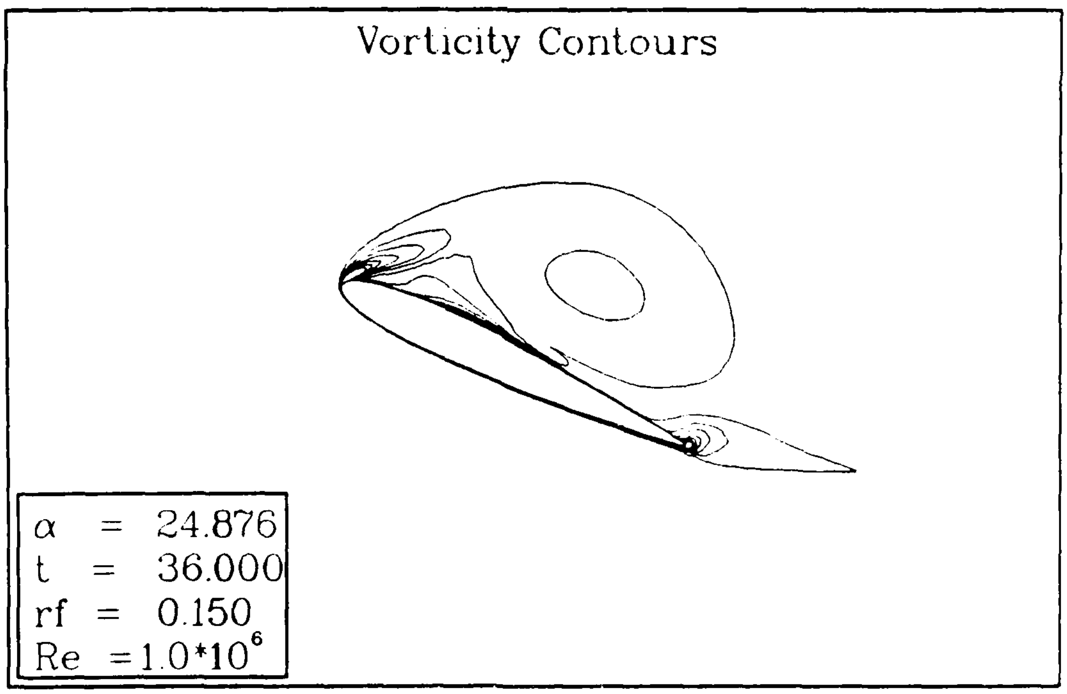

Vorticity Contours
α = 21.818
t = 28.000
rf = 0.150
Re = 1.0*10^6


Streamlines
α = 23.828
t = 32.000
rf = 0.150
Re =1.0*10^6

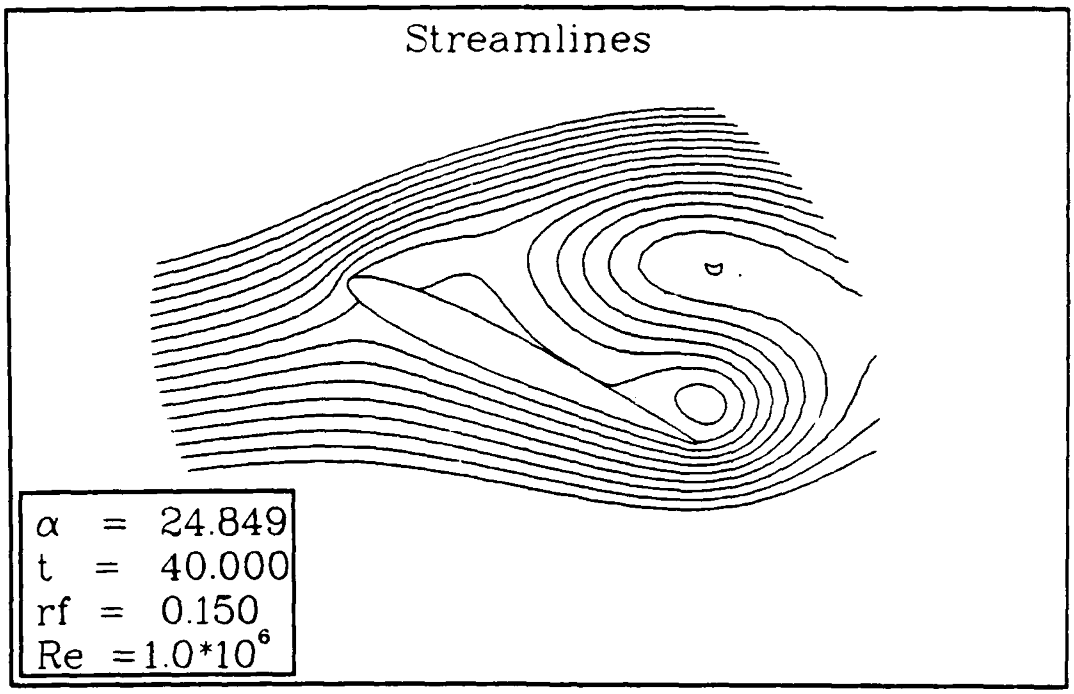

Vorticity Contours
α = 23.828
t = 32.000
rf = 0.150
Re =1.0*10^6

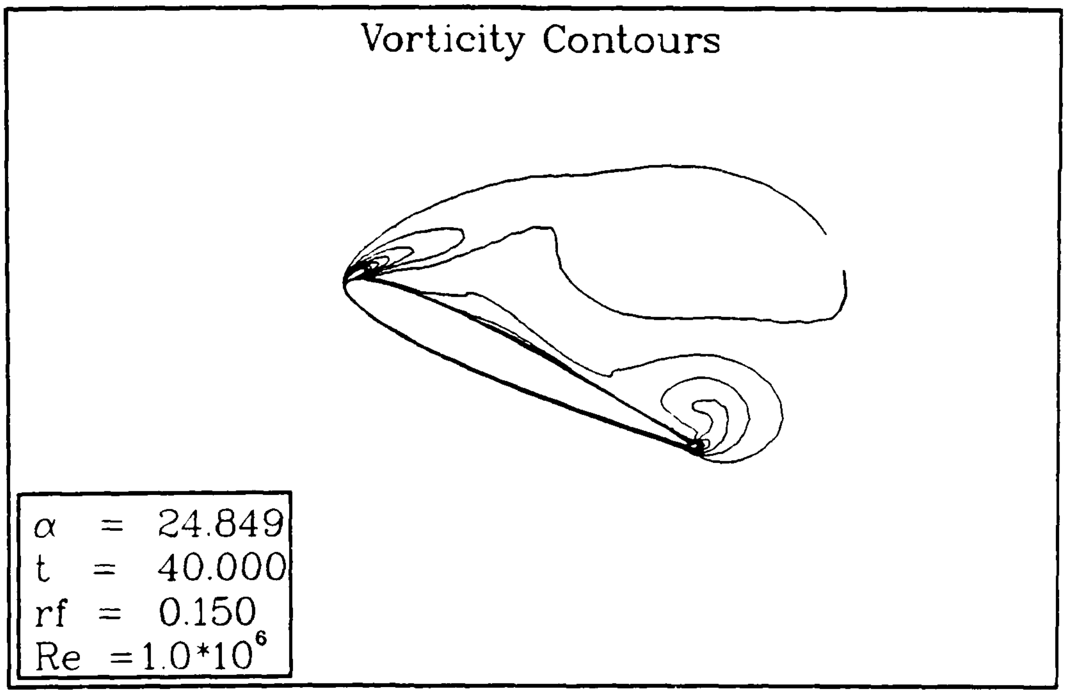


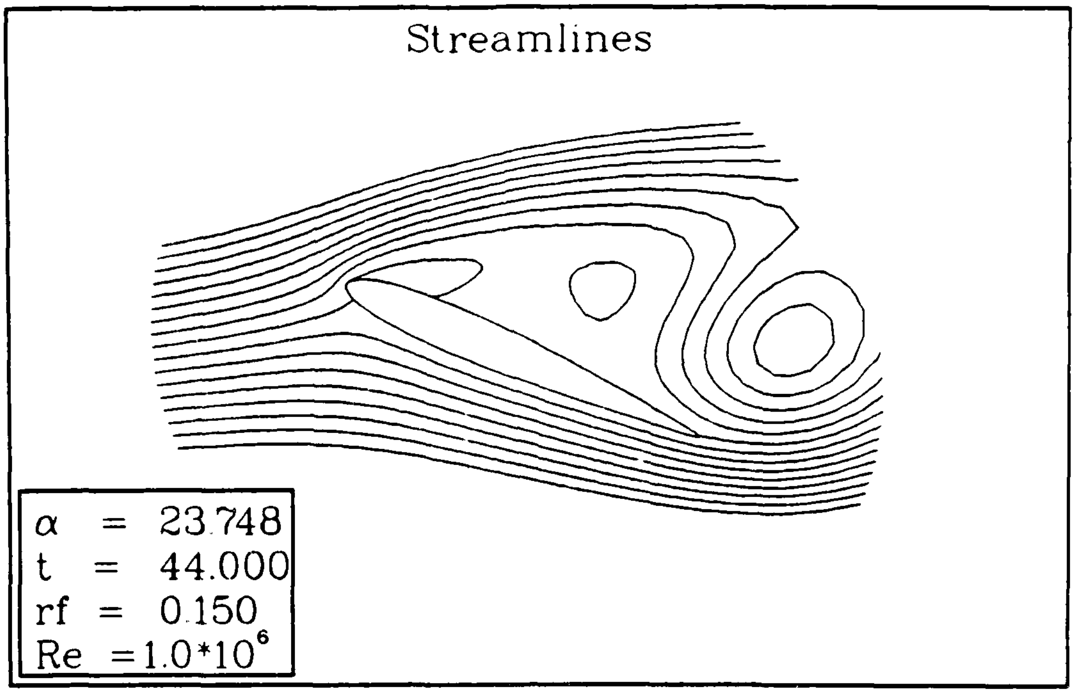
Streamlines
α = 24.876
t = 36.000
rf = 0.150
Re =1.0*10^6

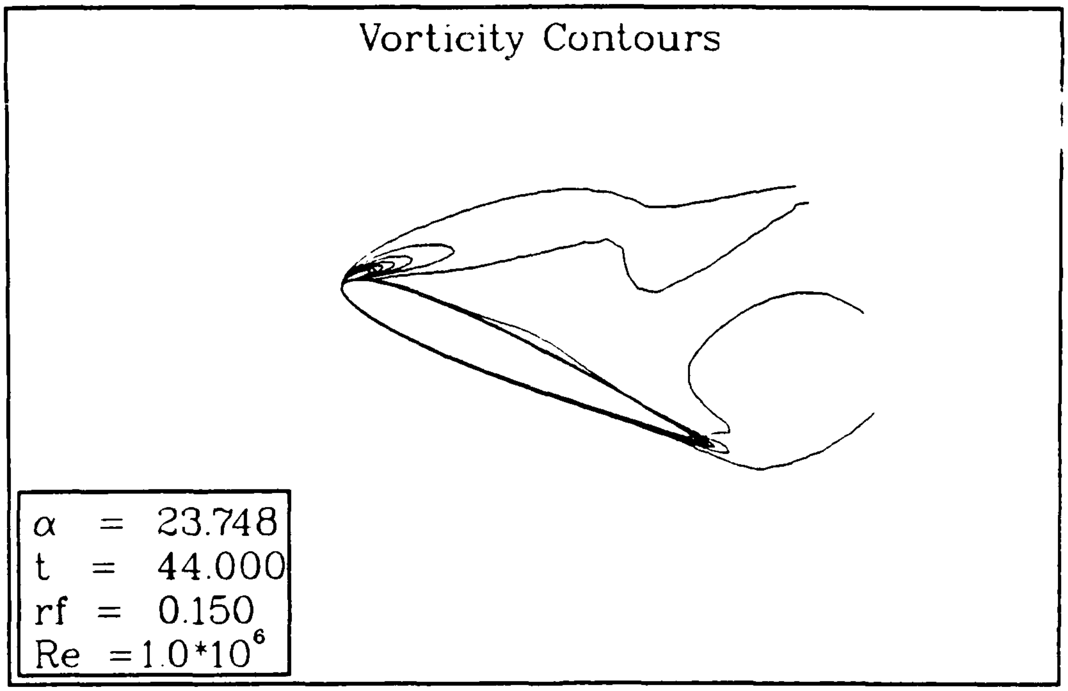


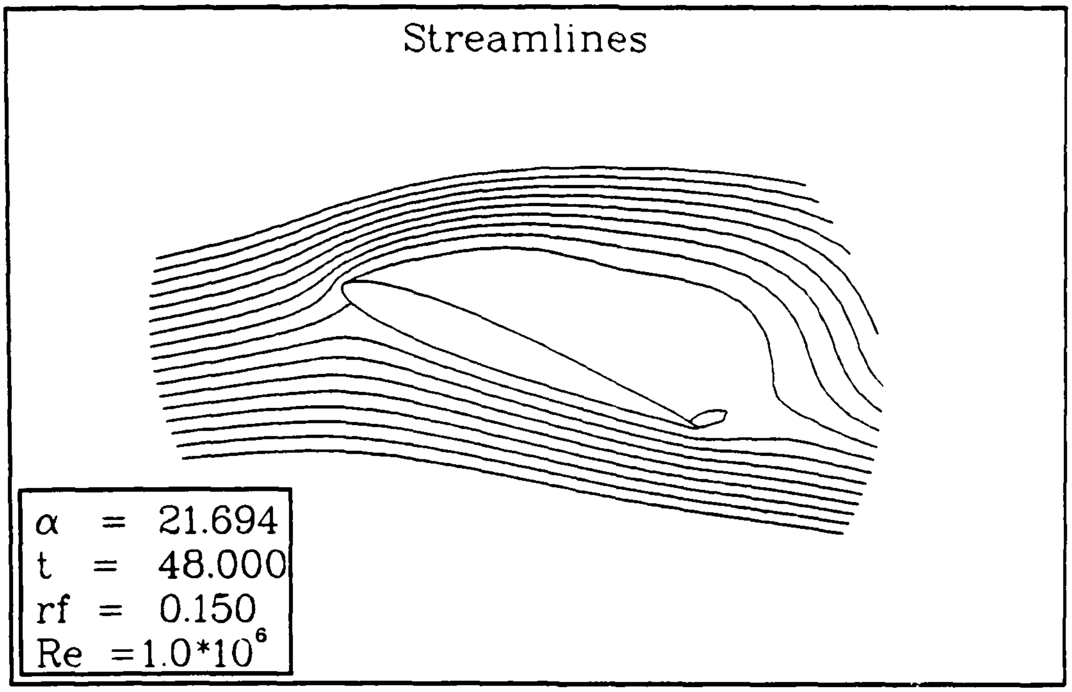
Vorticity Contours
α = 24.876
t = 36.000
rf = 0.150
Re =1.0*10^6

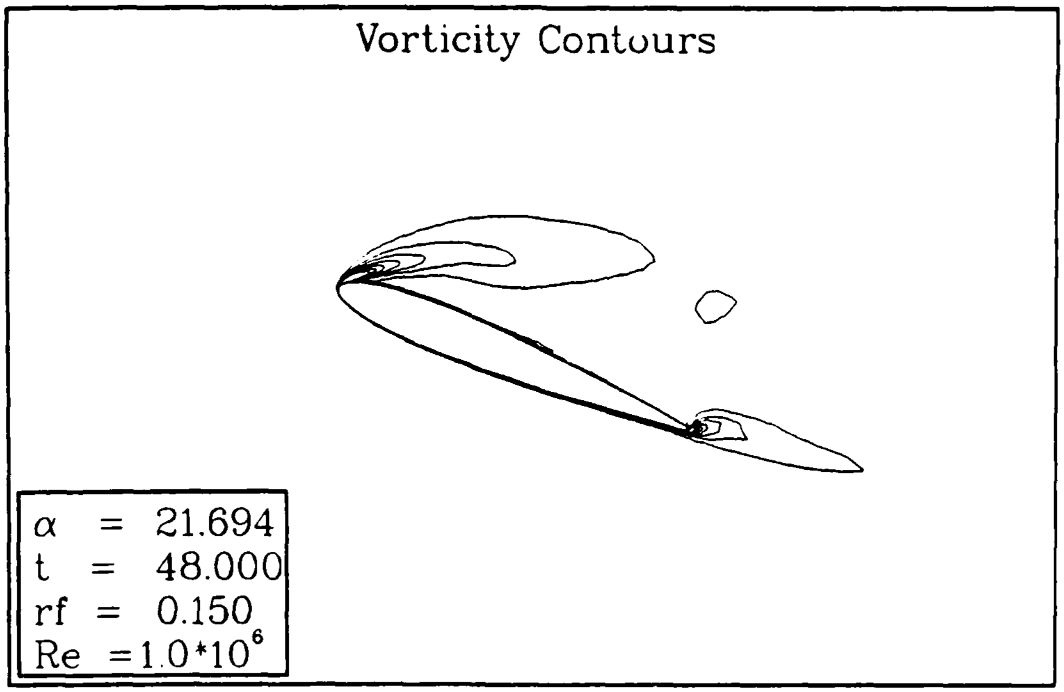

Streamlines
α = 24.849
t = 40.000
rf = 0.150
Re =1.0*10^6

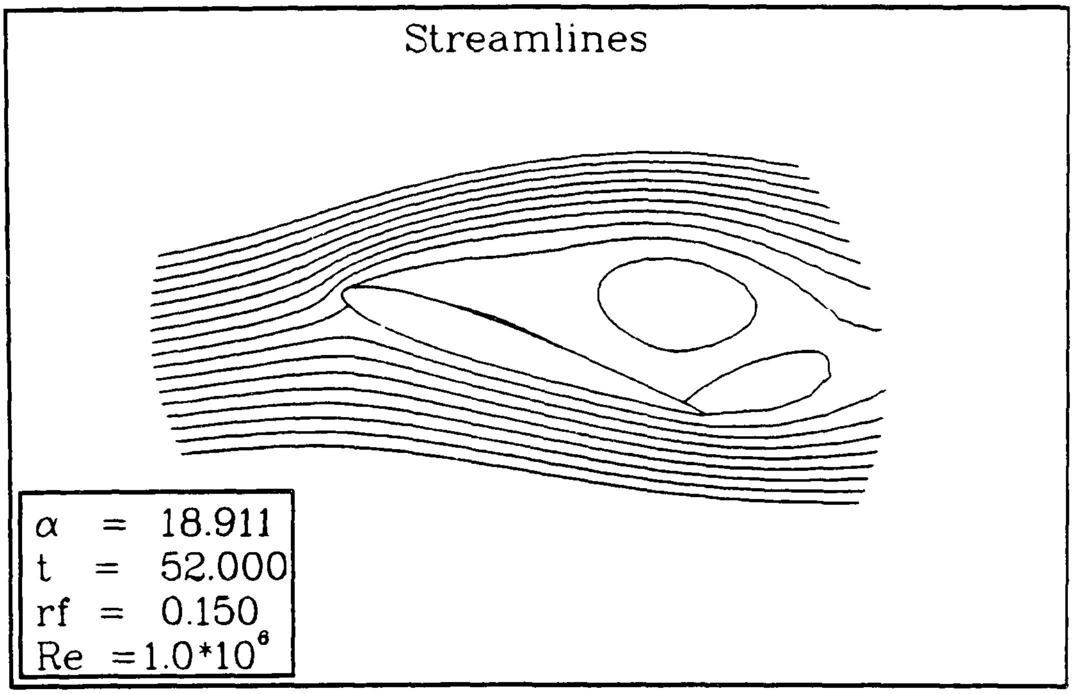

Vorticity Contours
α = 24.849
t = 40.000
rf = 0.150
Re =1.0*10^6

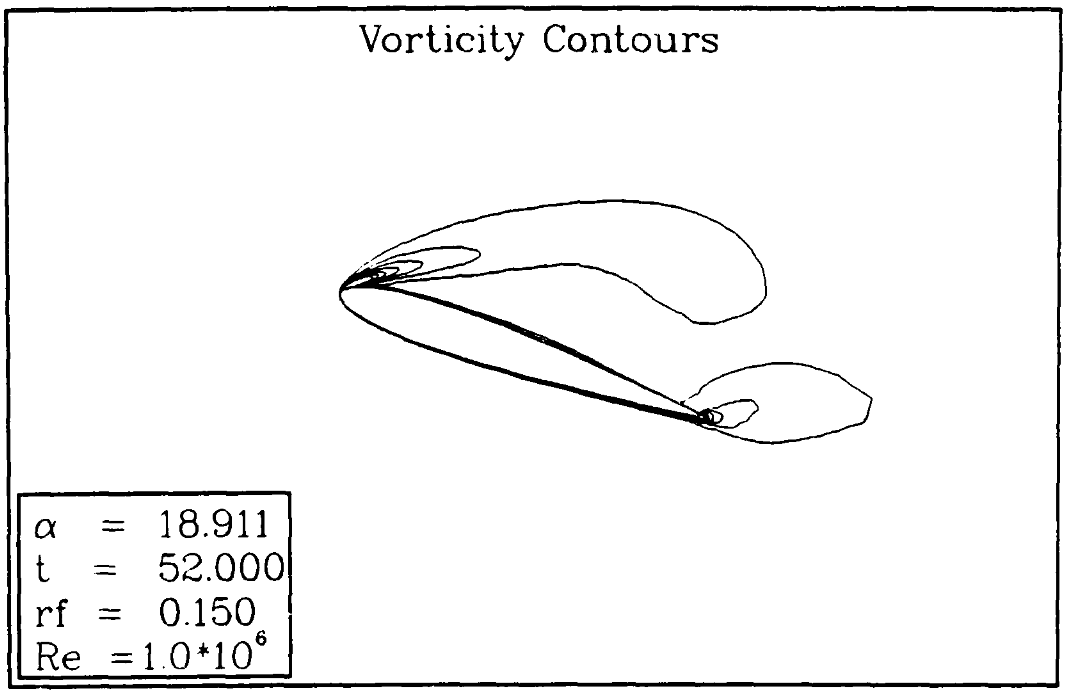

Streamlines
α = 23.748
t = 44.000
rf = 0.150
Re =1.0*10^6

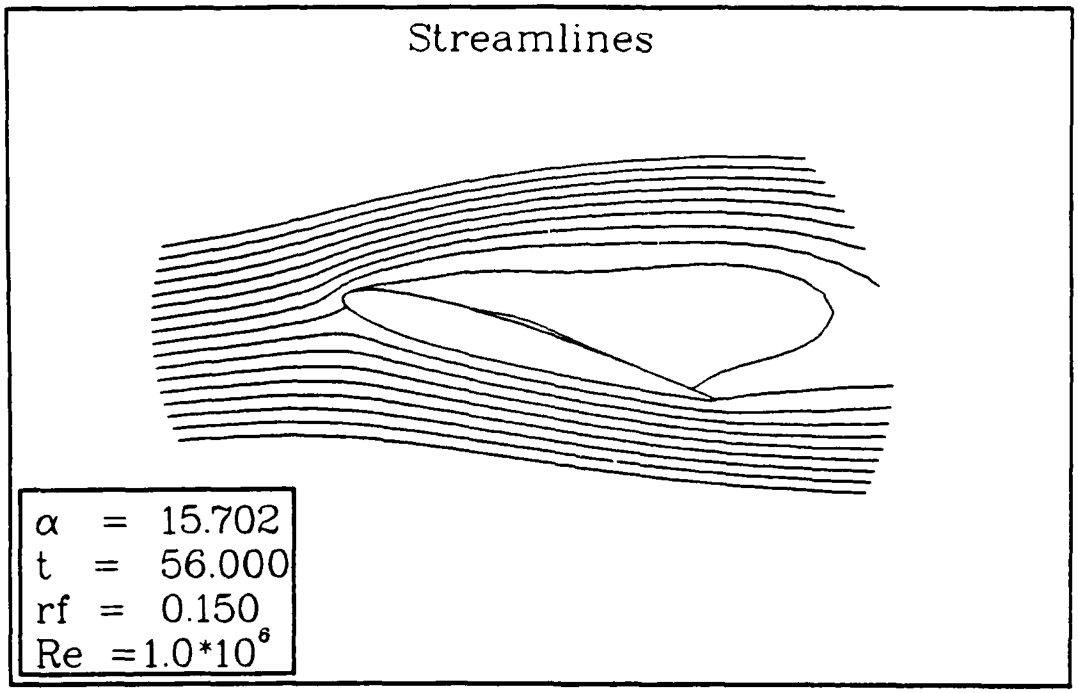

Vorticity Contours
α = 23.748
t = 44.000
rf = 0.150
Re =1.0*10^6

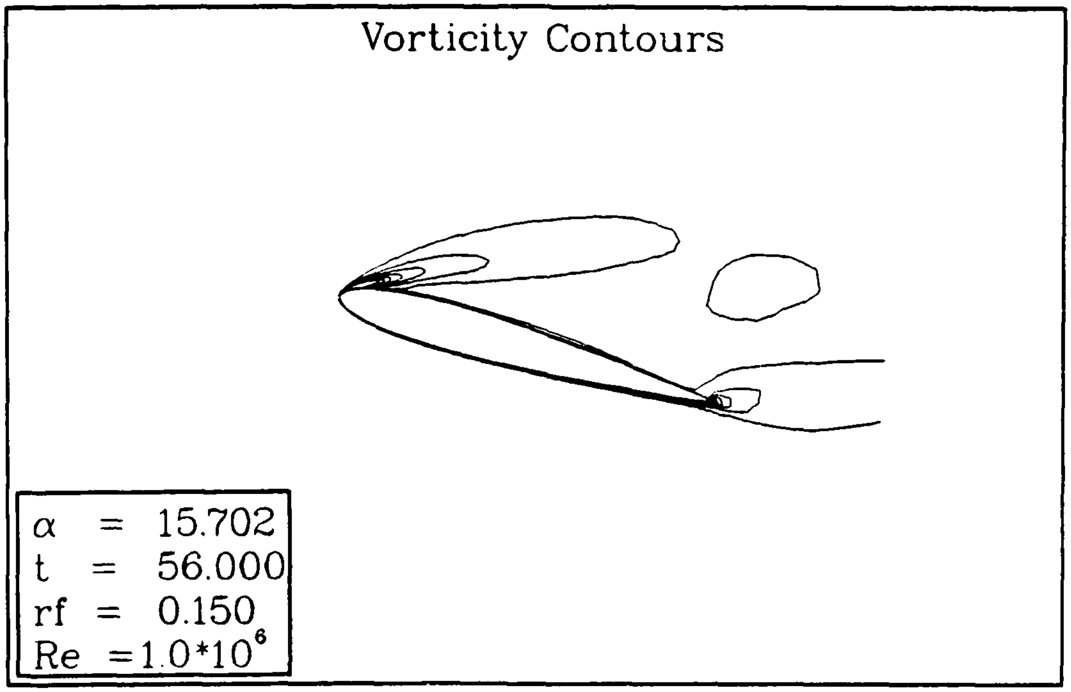

Streamlines
α = 21.694
t = 48.000
rf = 0.150
Re =1.0*10^6

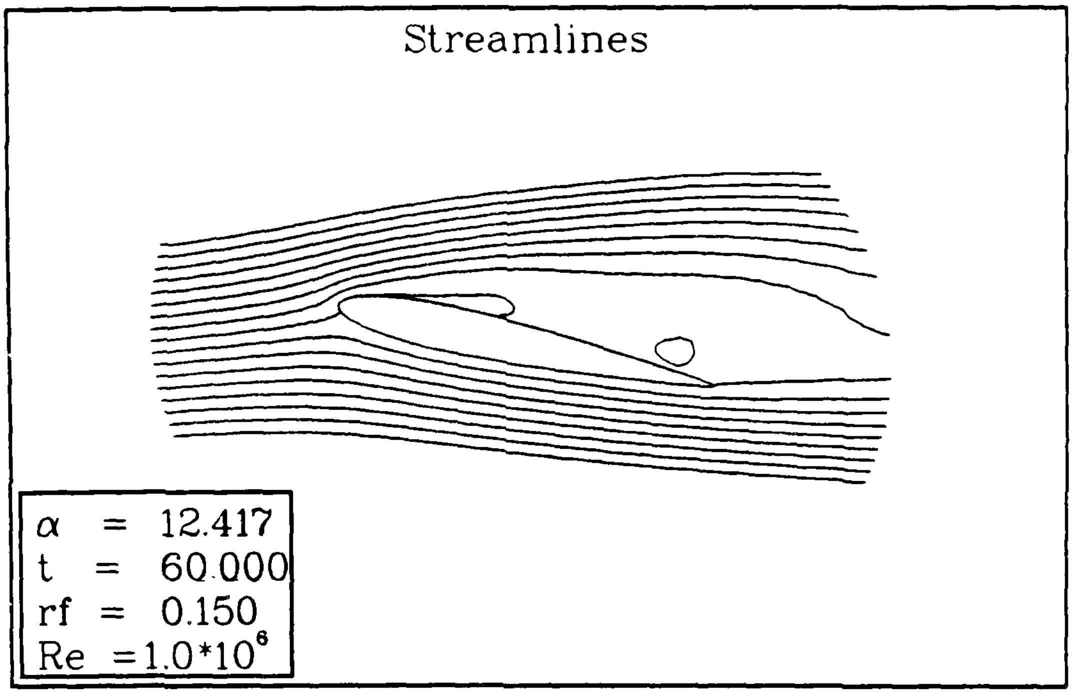

Vorticity Contours
α = 21.694
t = 48.000
rf = 0.150
Re =1.0*10^6

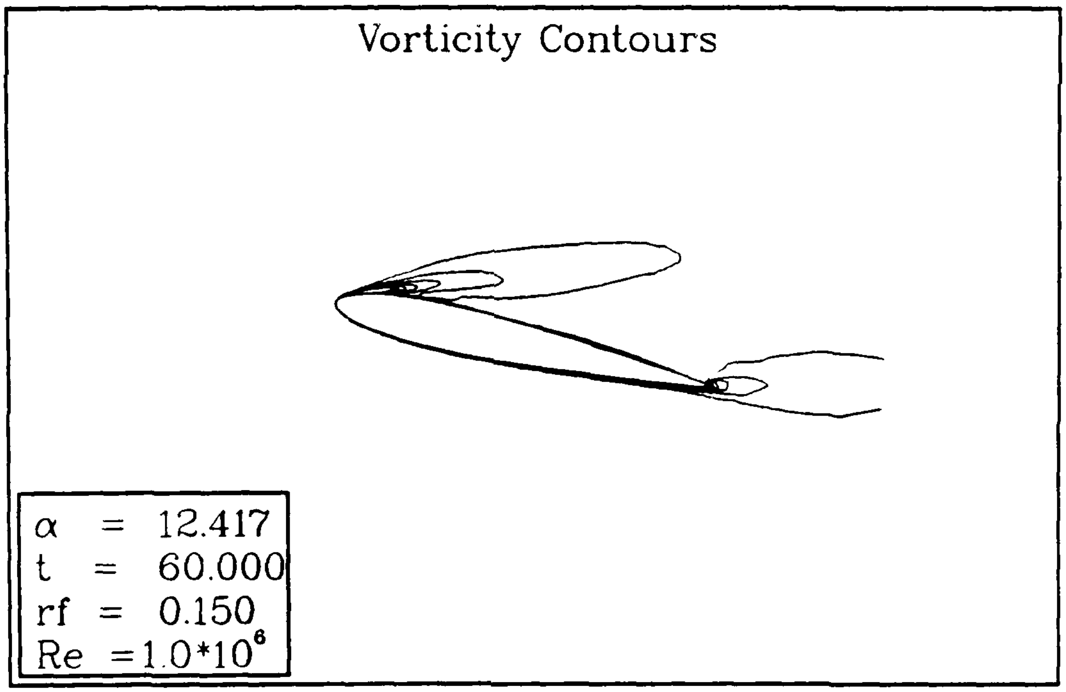

Streamlines
α = 18.911
t = 52.000
rf = 0.150
Re =1.0*10^6

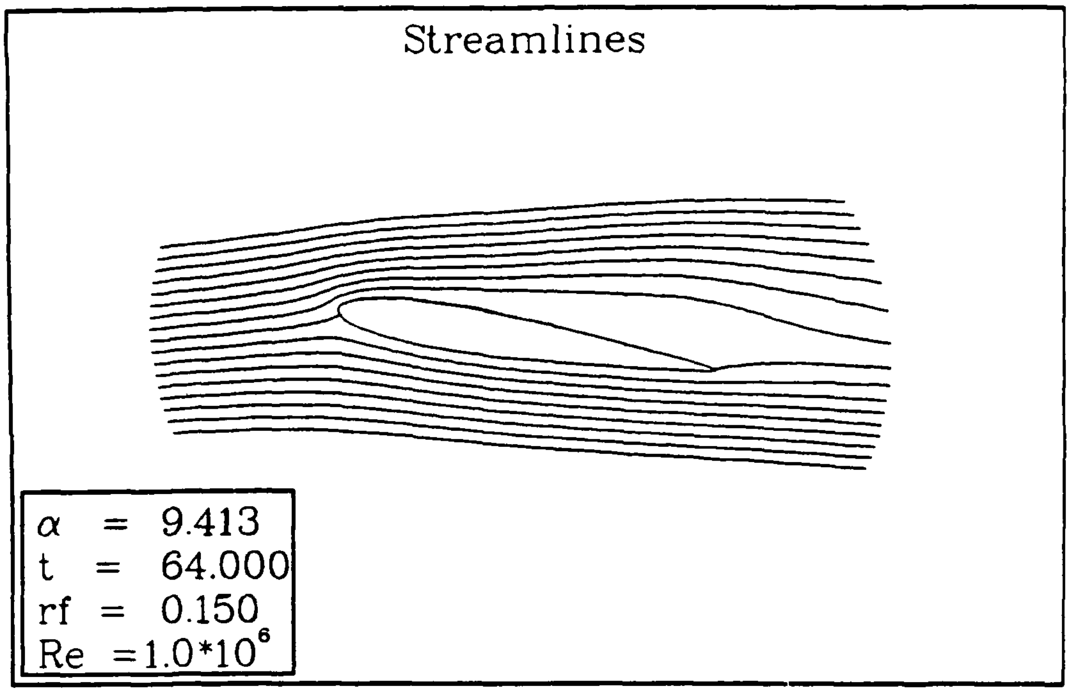

Vorticity Contours
α = 18.911
t = 52.000
rf = 0.150
Re =1.0*10^6

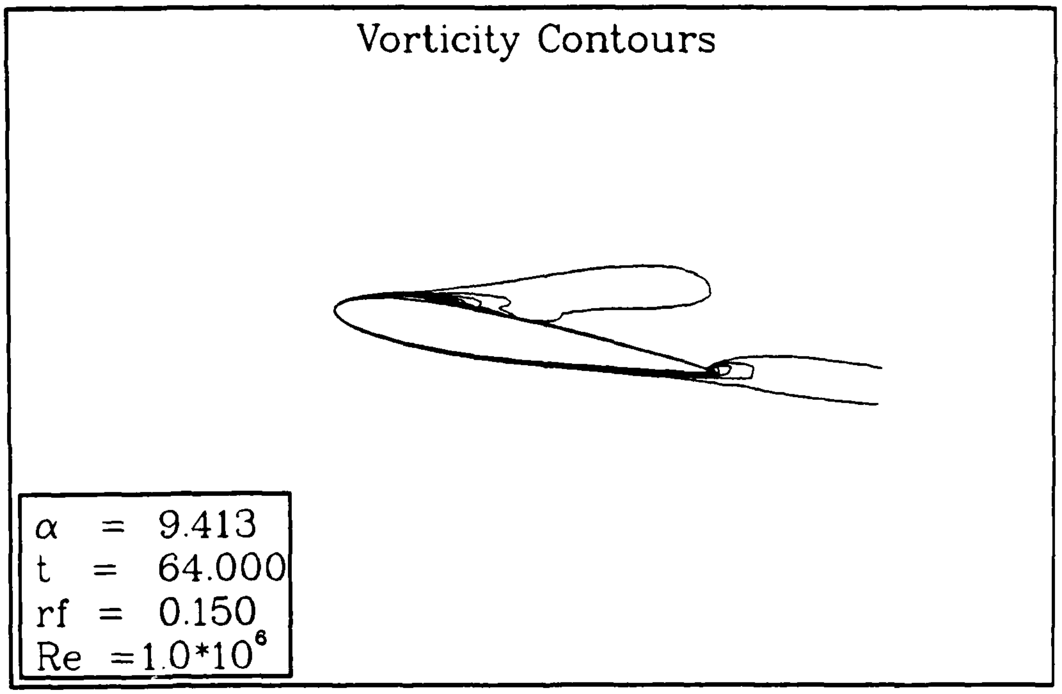

Streamlines
α = 15.702
t = 56.000
rf = 0.150
Re =1.0*10^6

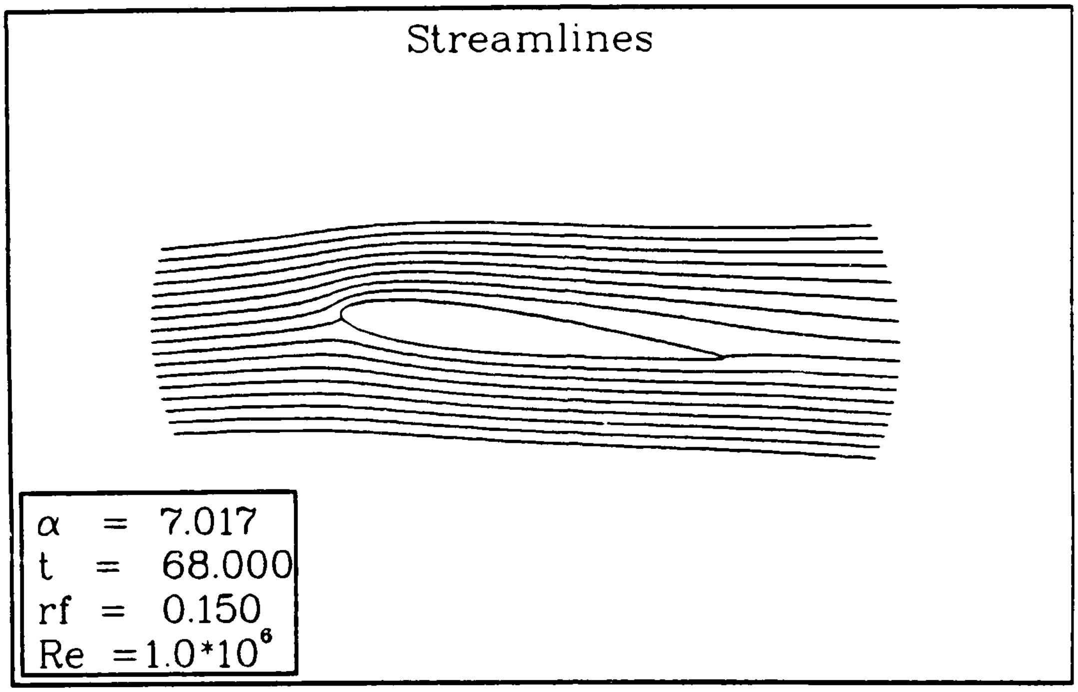

Vorticity Contours
α = 15.702
t = 56.000
rf = 0.150
Re =1.0*10^6

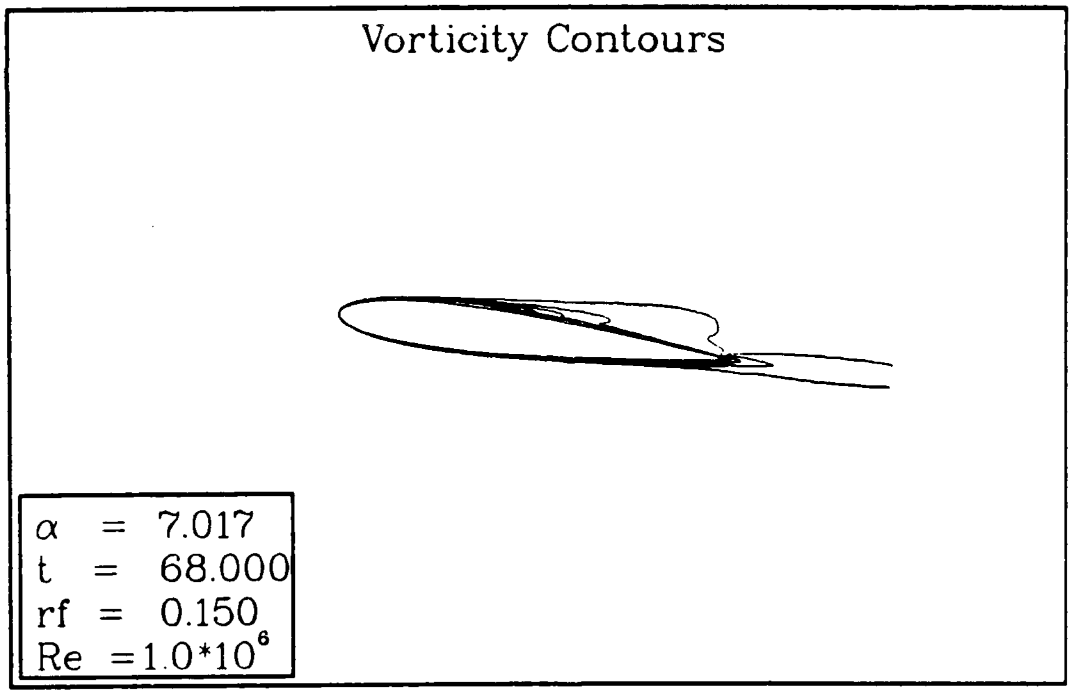

Streamlines
α = 12.417
t = 60.000
rf = 0.150
Re =1.0*10^6

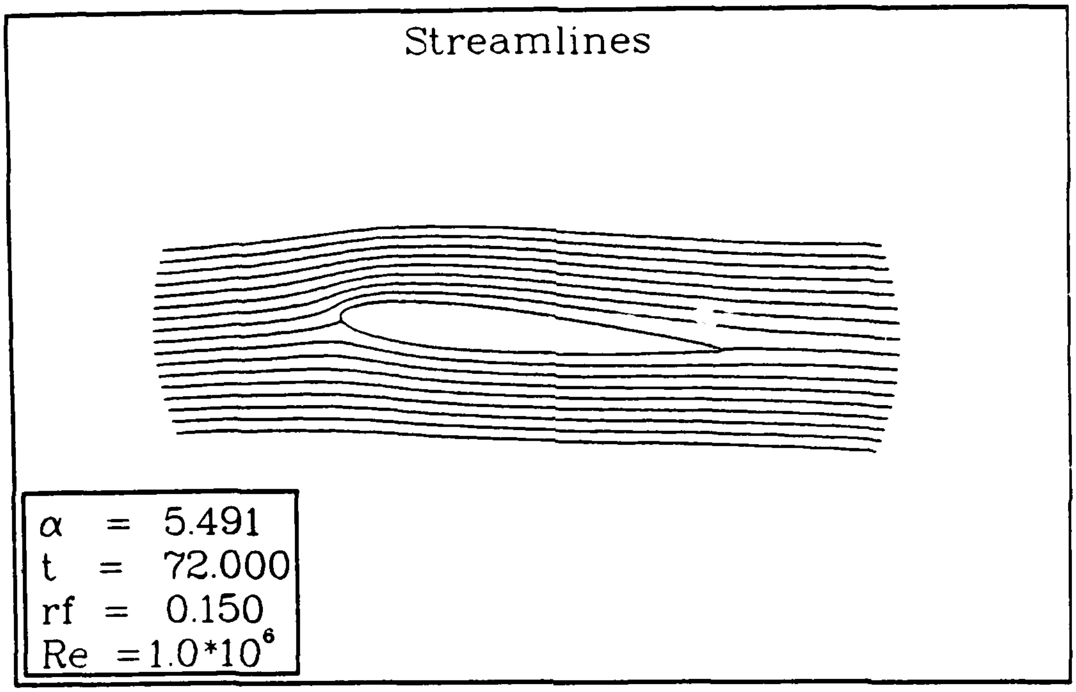

Vorticity Contours
α = 12.417
t = 60.000
rf = 0.150
Re =1.0*10^6


Streamlines
α = 9.413
t = 64.000
rf = 0.150
Re =1.0*10^6

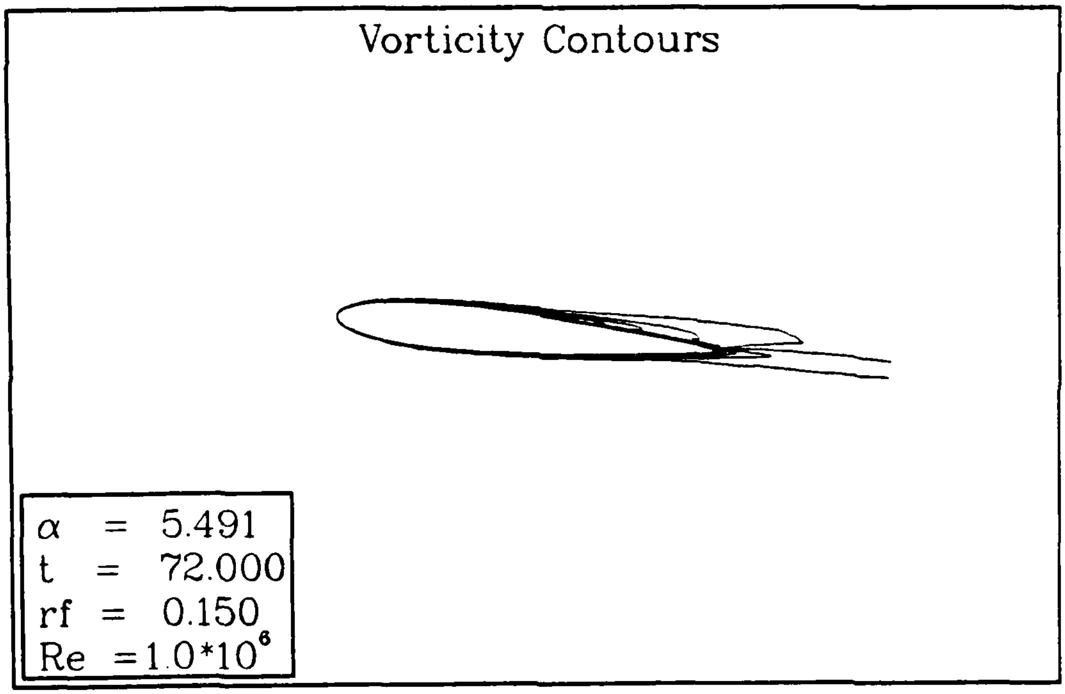

Vorticity Contours
α = 9.413
t = 64.000
rf = 0.150
Re =1.0*10^6


Streamlines
α = 7.017
t = 68.000
rf = 0.150
Re =1.0*10^6

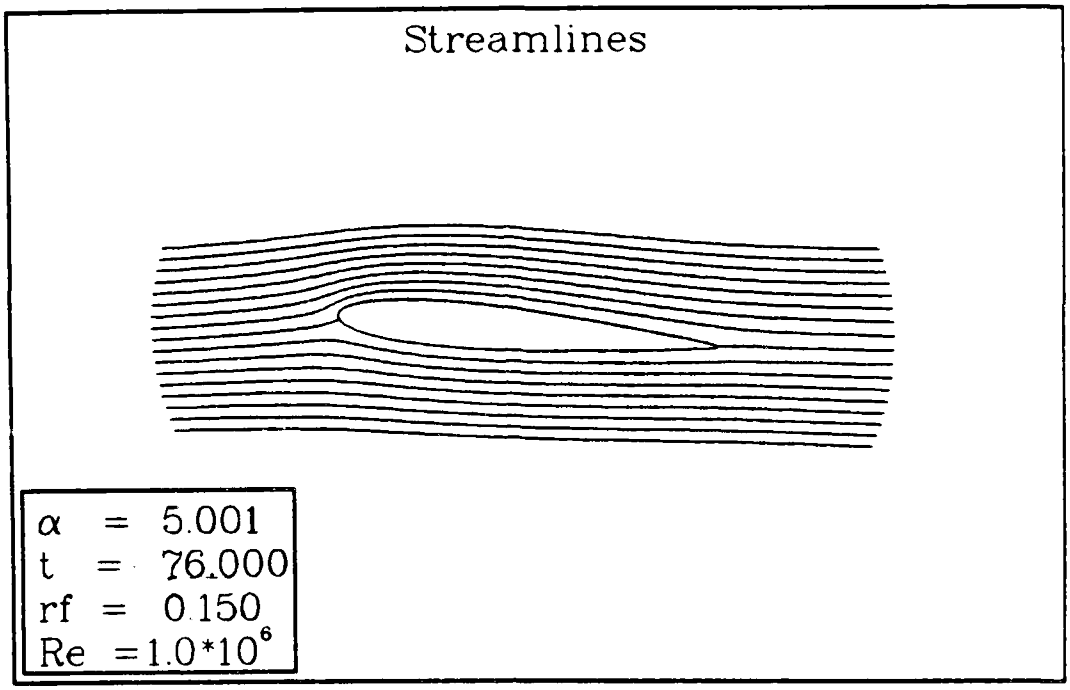

Vorticity Contours
α = 7.017
t = 68.000
rf = 0.150
Re =1.0*10^6

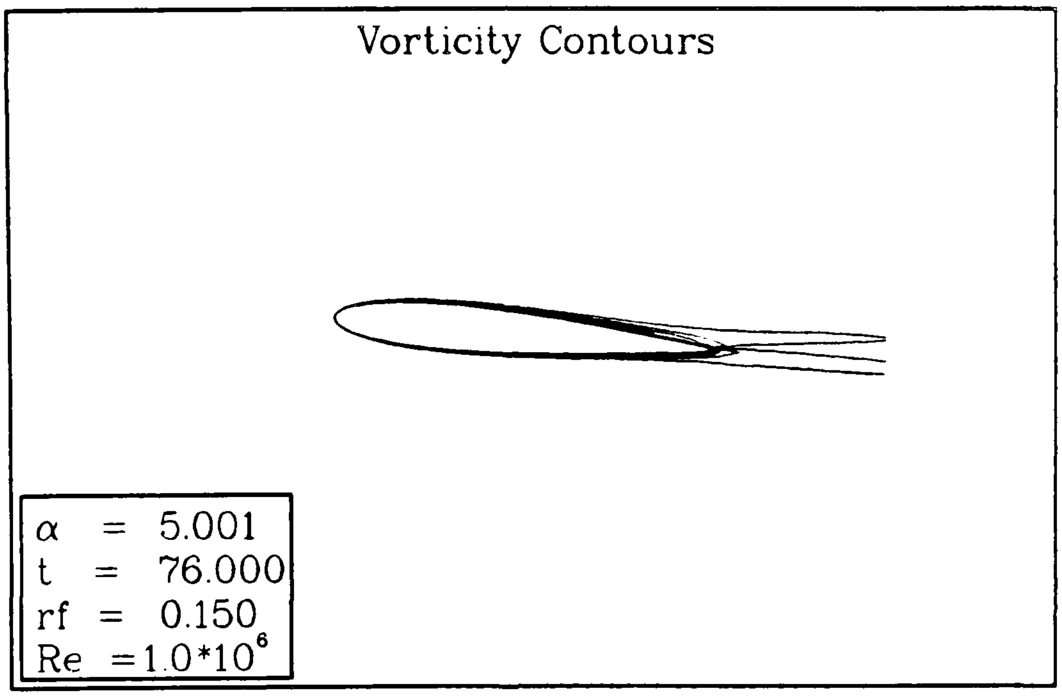


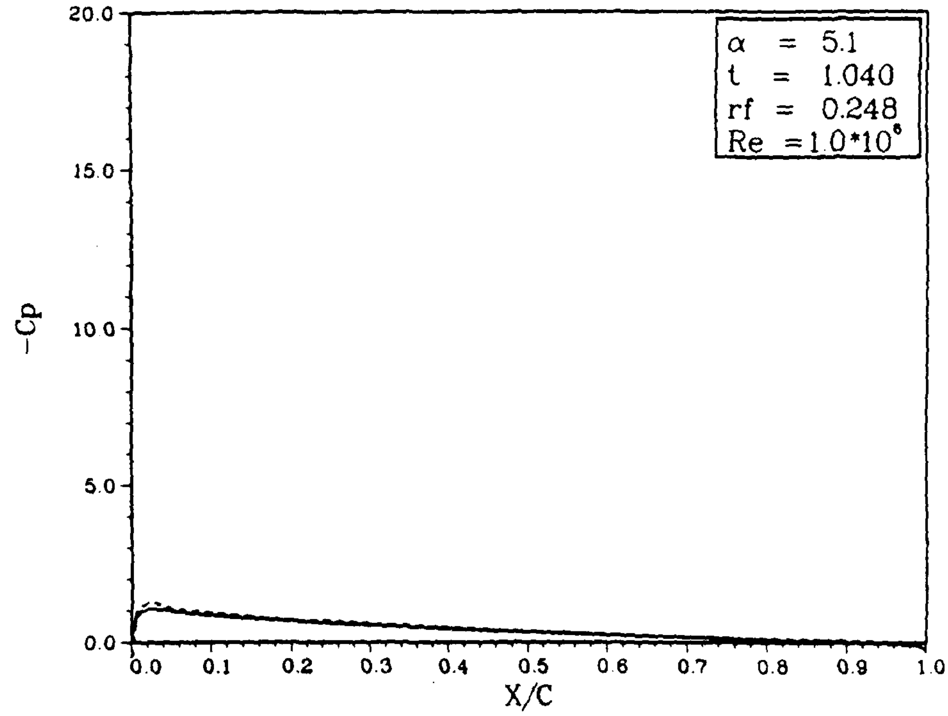
Streamlines
α = 5.491
t = 72.000
rf = 0.150
Re = 1.0*10^6

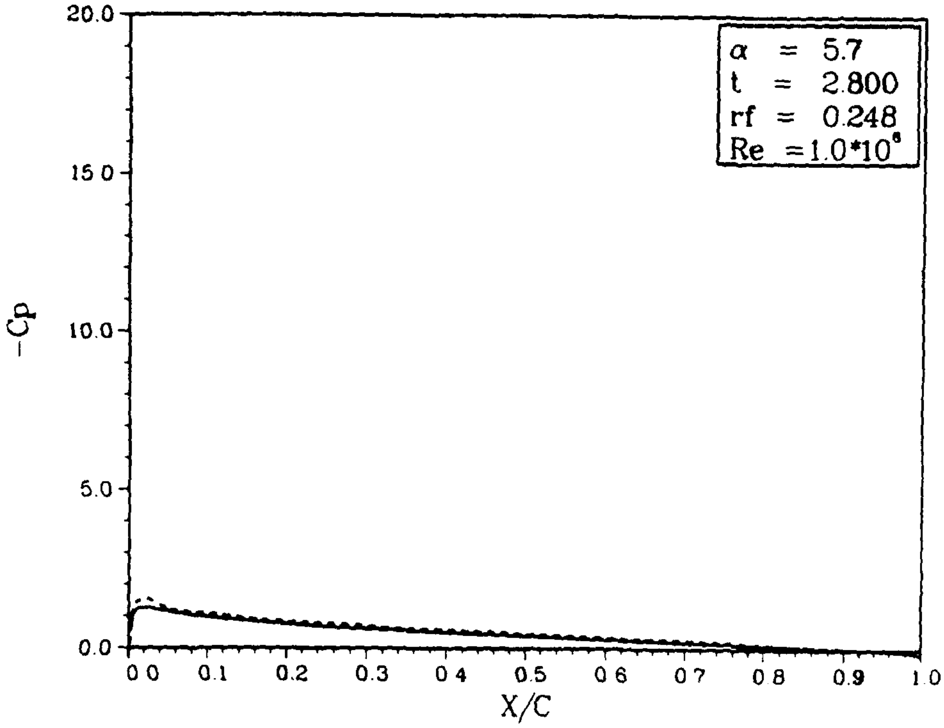


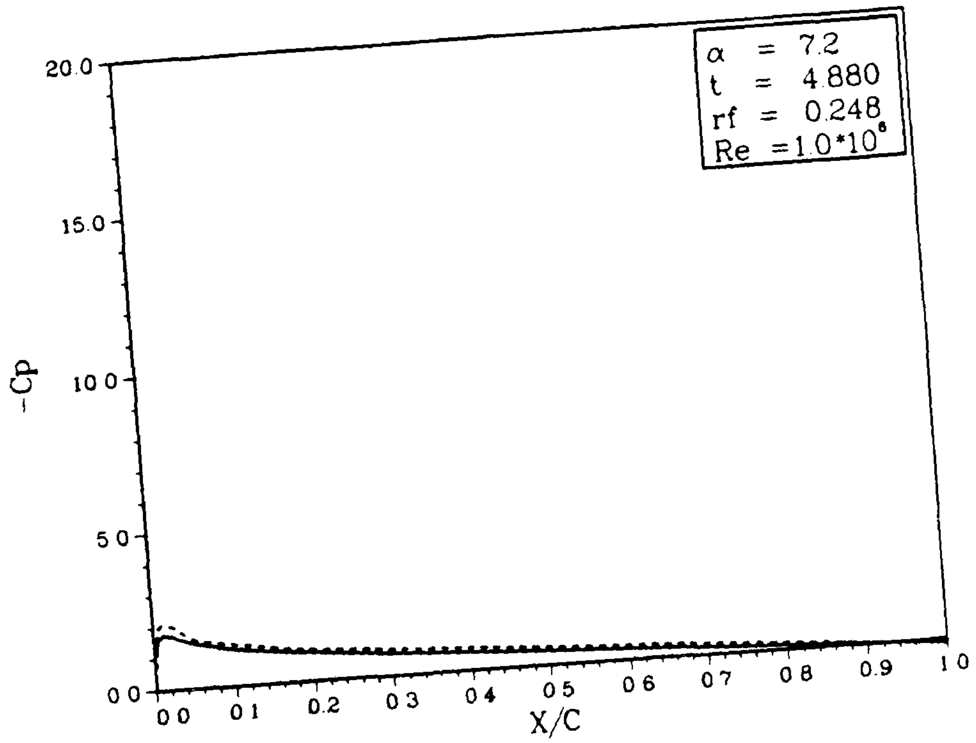
Vorticity Contours
α = 5.491
t = 72.000
rf = 0.150
Re = 1.0*10^6

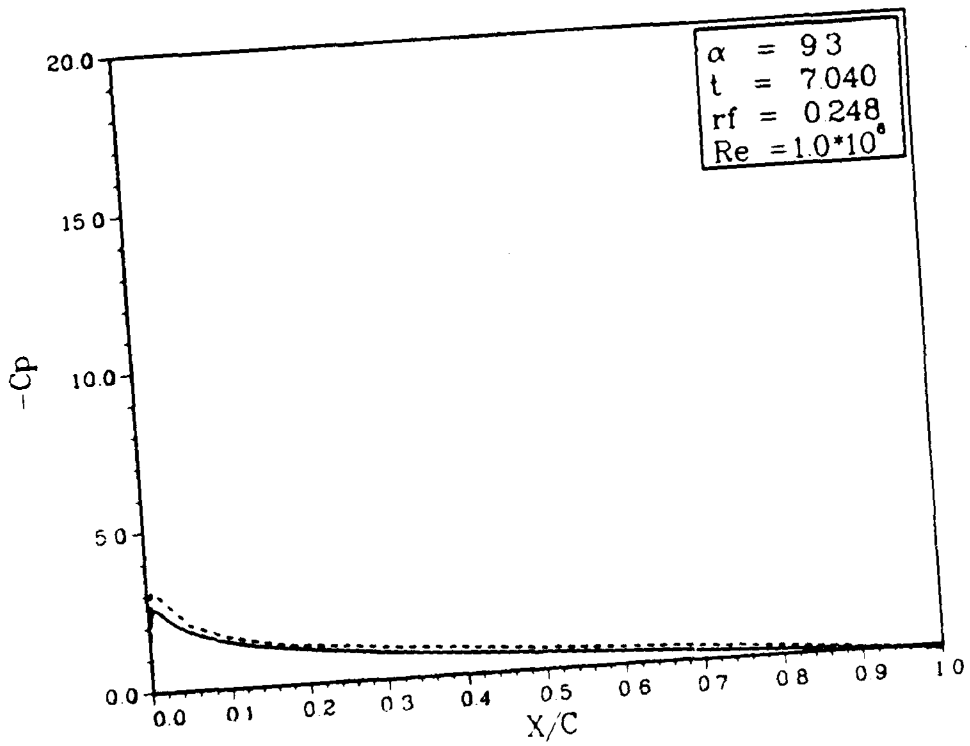

Streamlines
α = 5.001
t = 76.000
rf = 0.150
Re =1.0*10^6

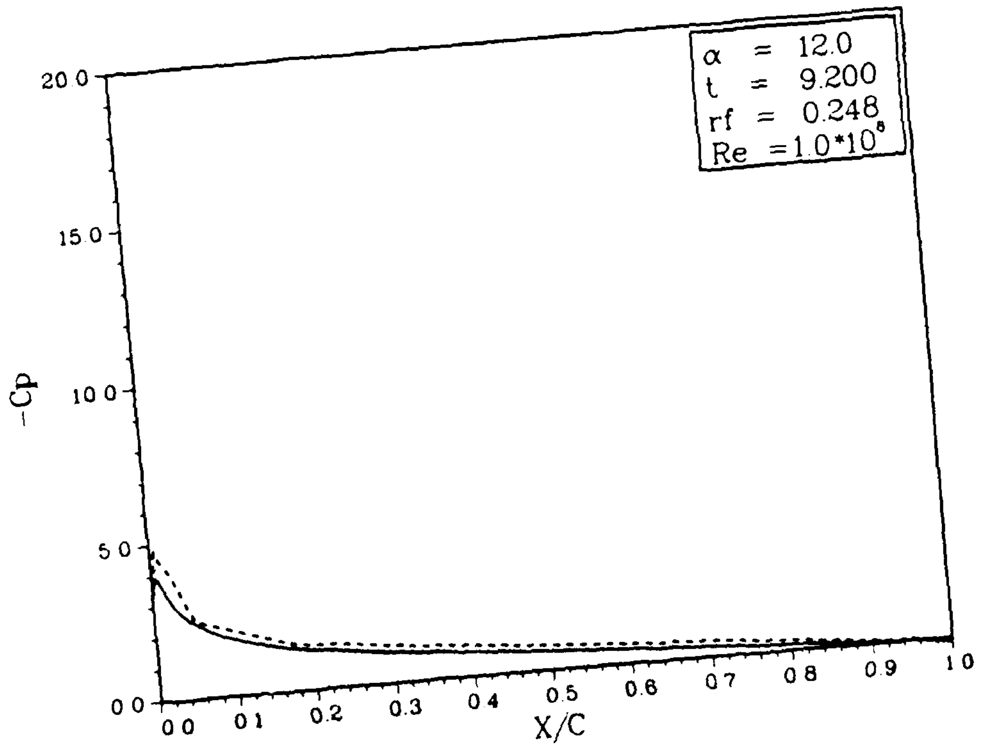

Vorticity Contours
α = 5.001
t = 76.000
rf = 0.150
Re =1.0*10^6

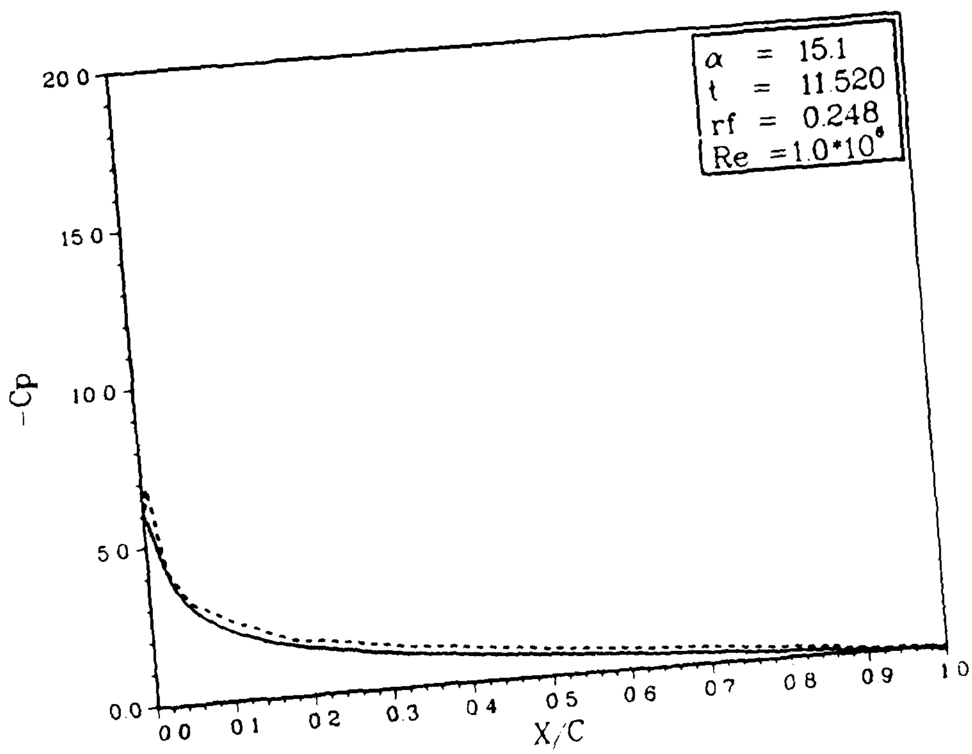

20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C
α = 5.1
t = 1.040
rf = 0.248
Re =1.0*10^6

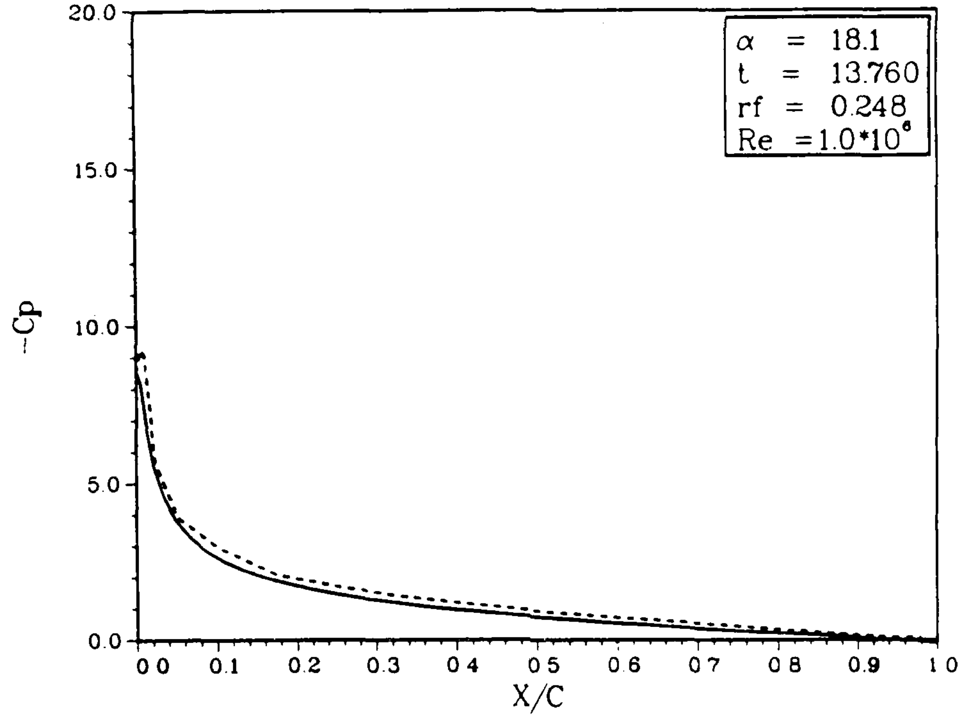

20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C
α = 5.7
t = 2.800
rf = 0.248
Re =1.0*10^6


α = 7.2
t = 4.880
rf = 0.248
Re =1.0*10^6
-Cp
X/C

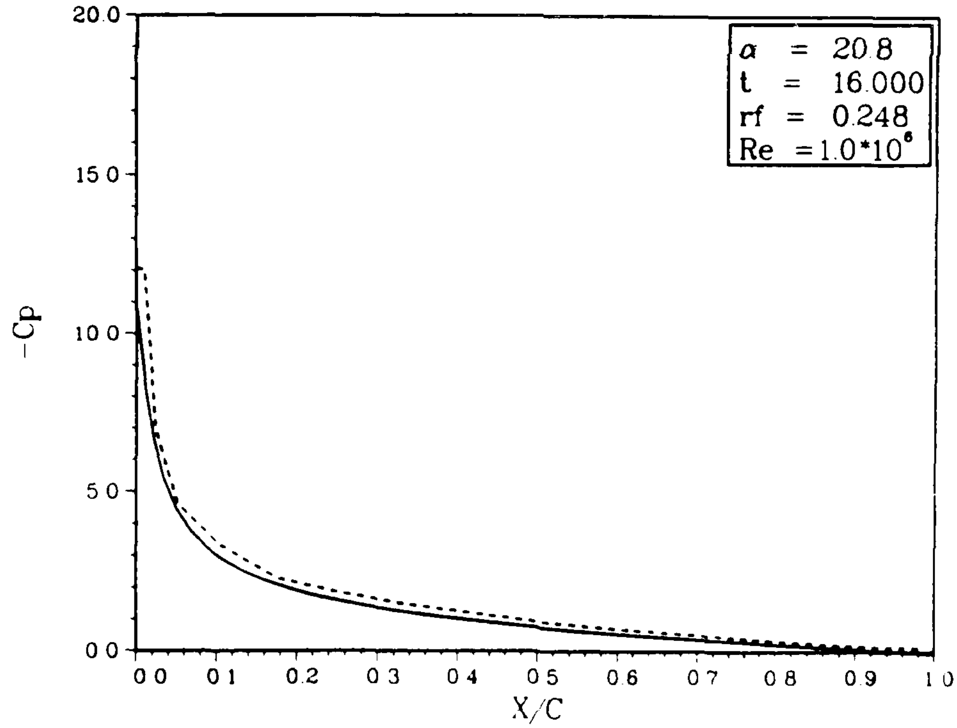

α = 9.3
t = 7.040
rf = 0.248
Re =1.0*10^6
-Cp
X/C


α = 12.0
t = 9.200
rf = 0.248
Re = 1.0*10^6
-Cp
X/C

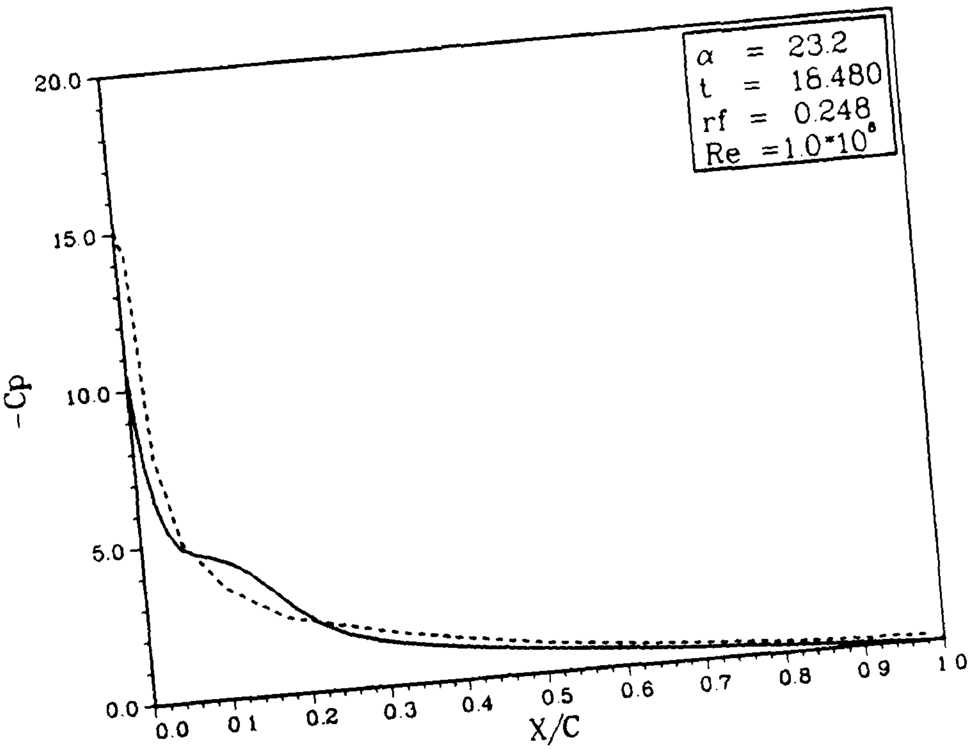

α = 15.1
t = 11.520
rf = 0.248
Re = 1.0*10^6
-Cp
X/C



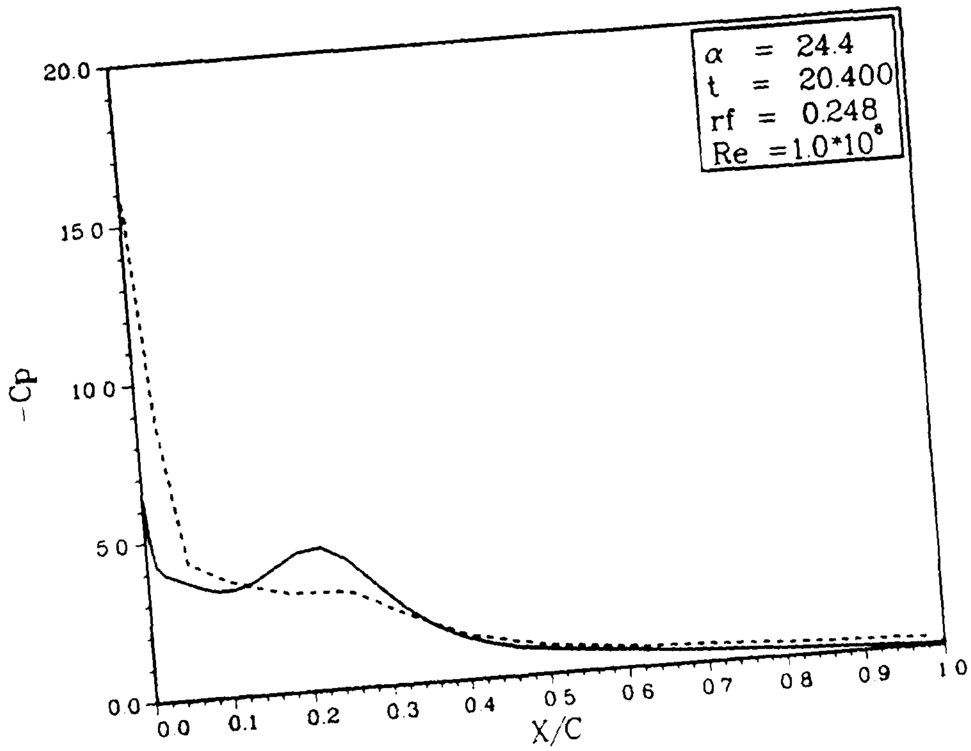
α = 18.1
t = 13.760
rf = 0.248
Re =1.0*10^6
-Cp
X/C



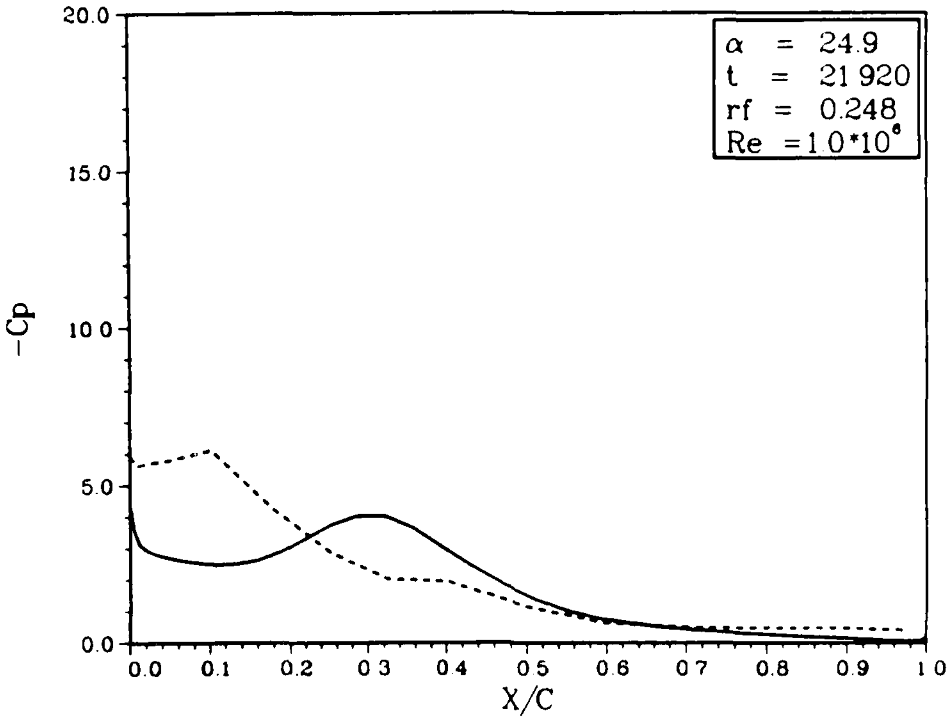
α = 20.8
t = 16.000
rf = 0.248
Re =1.0*10^6
-Cp
X/C



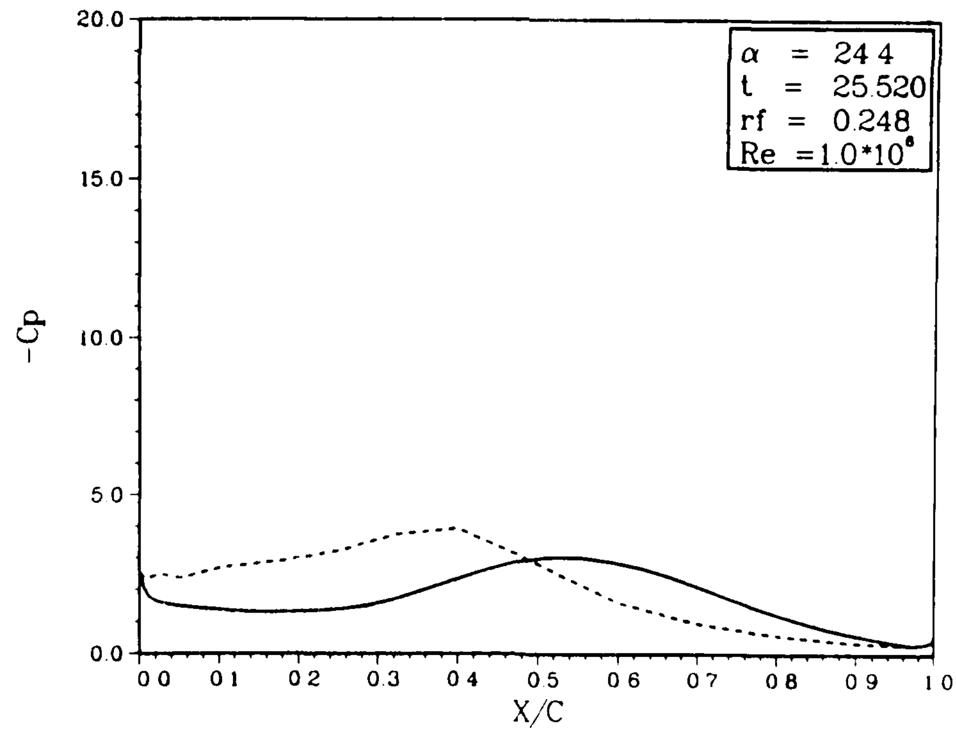
α = 23.2
t = 16.480
rf = 0.248
Re =1.0*10^6
-Cp
X/C



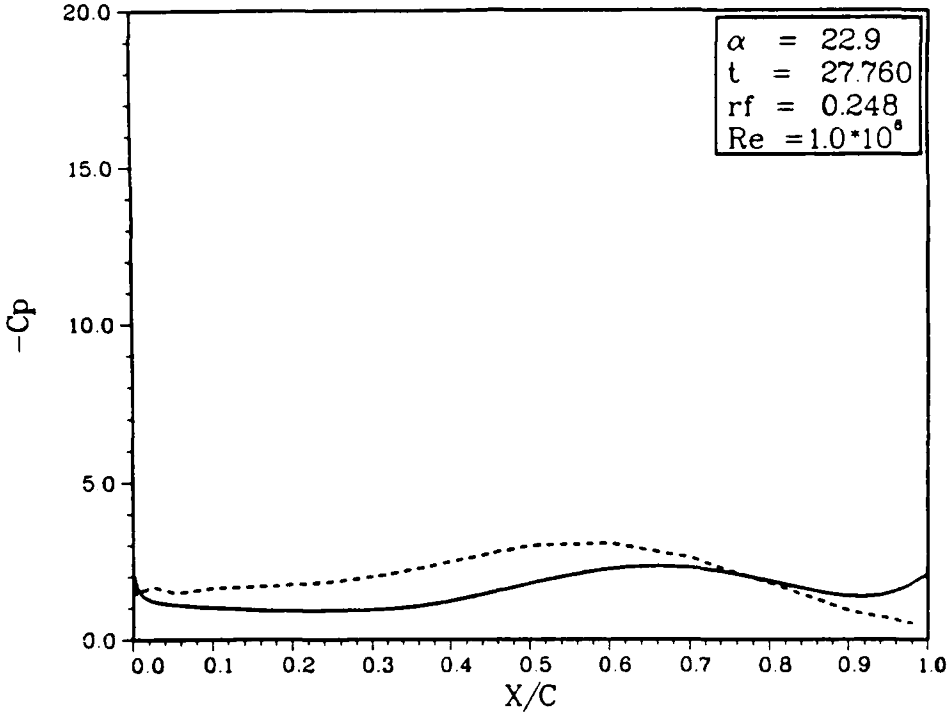
α = 24.4
t = 20.400
rf = 0.248
Re =1.0*10^6
-Cp
X/C

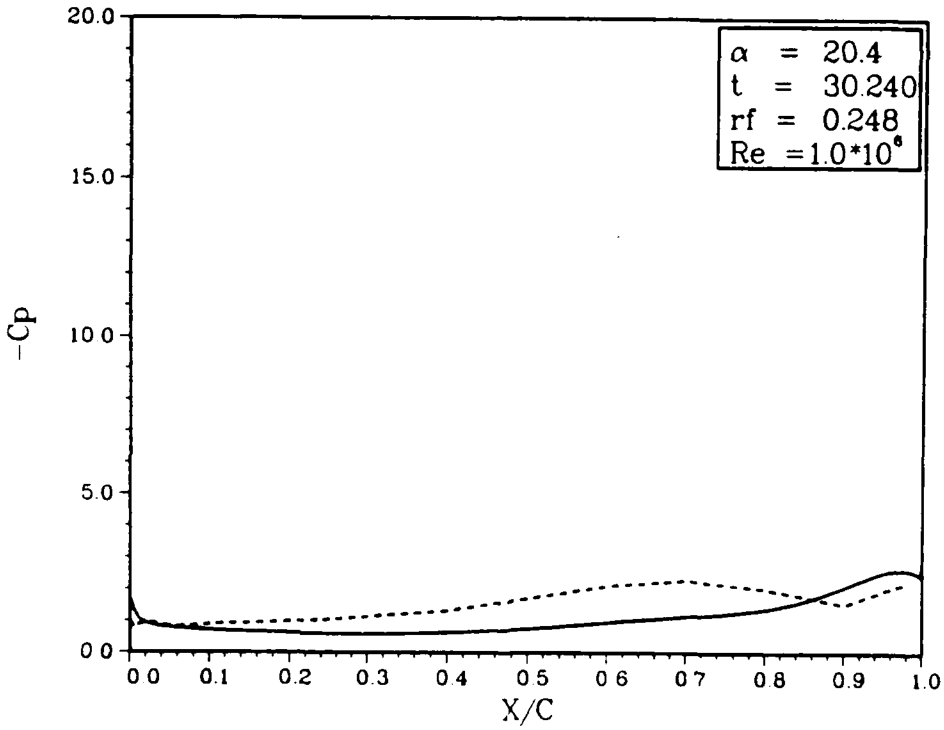


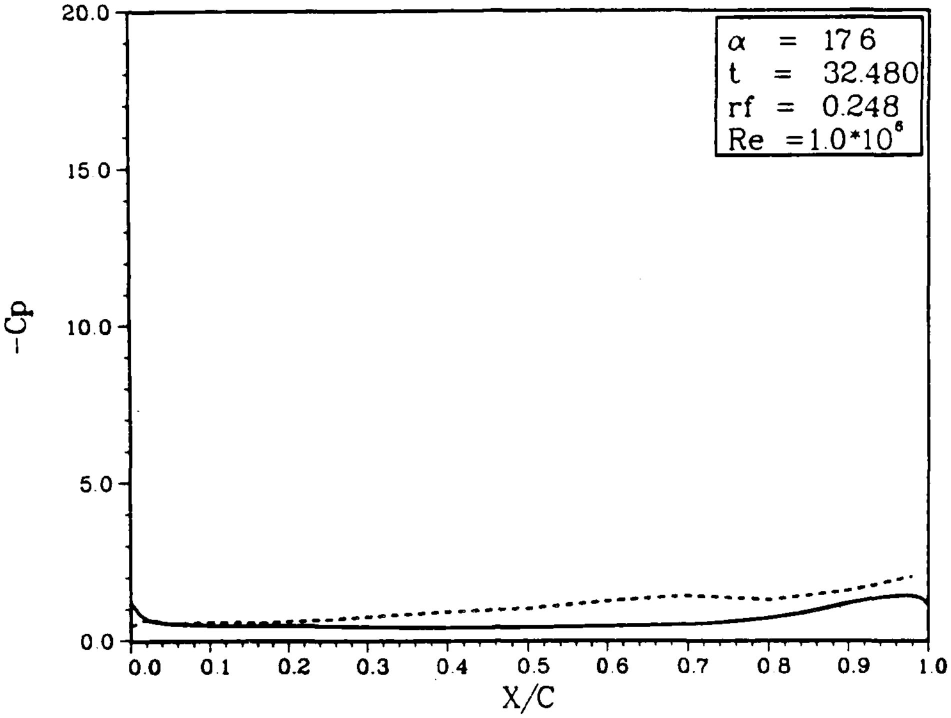
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C
α = 24.9
t = 21.920
rf = 0.248
Re = 1.0*10^6

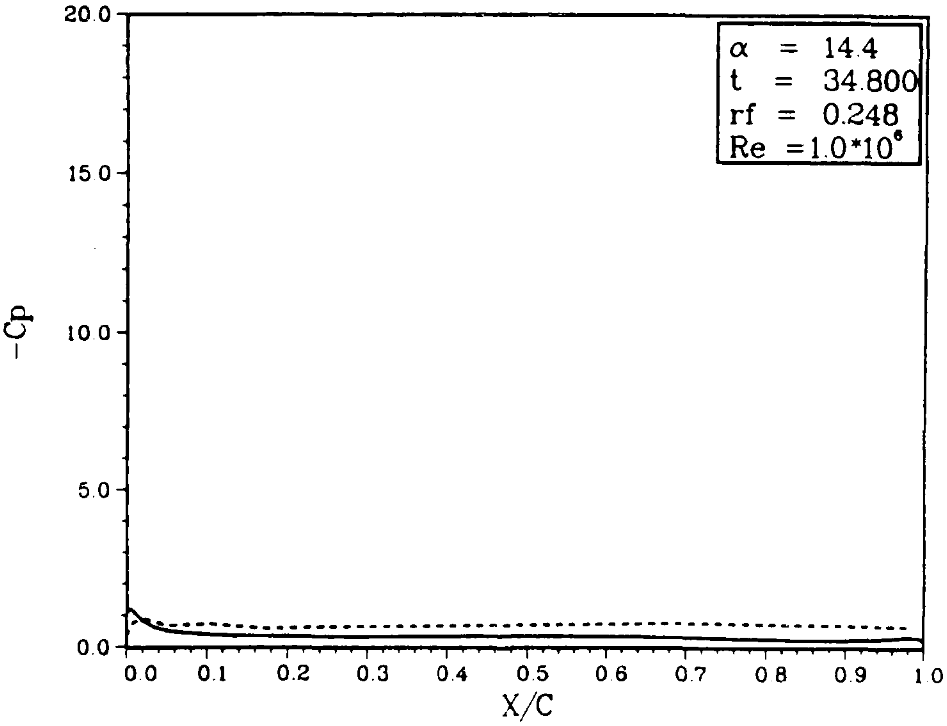


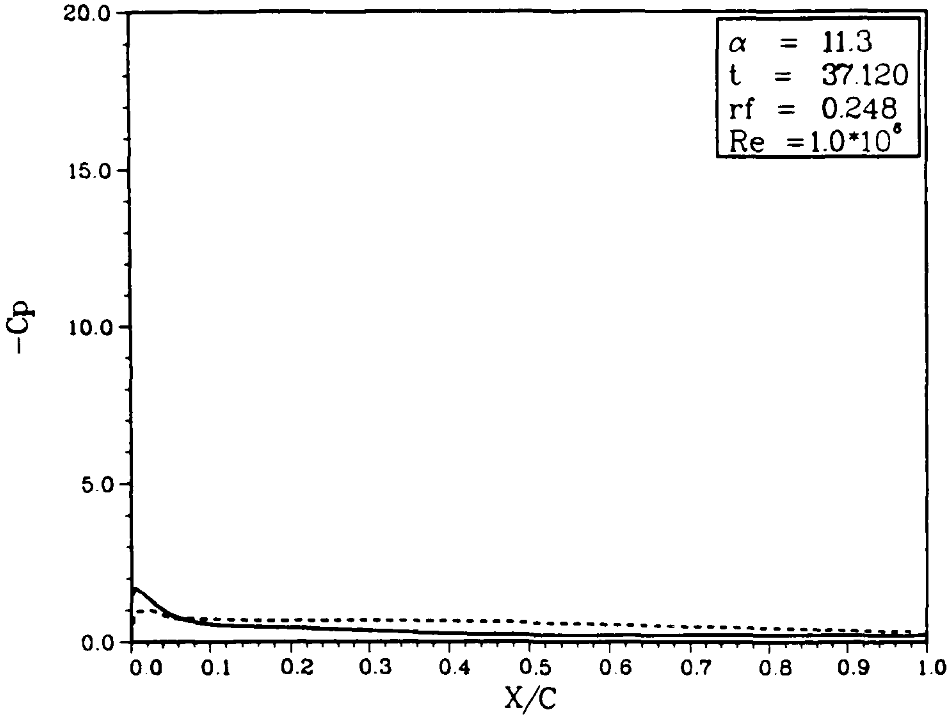
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C
α = 24.4
t = 25.520
rf = 0.248
Re = 1.0*10^6

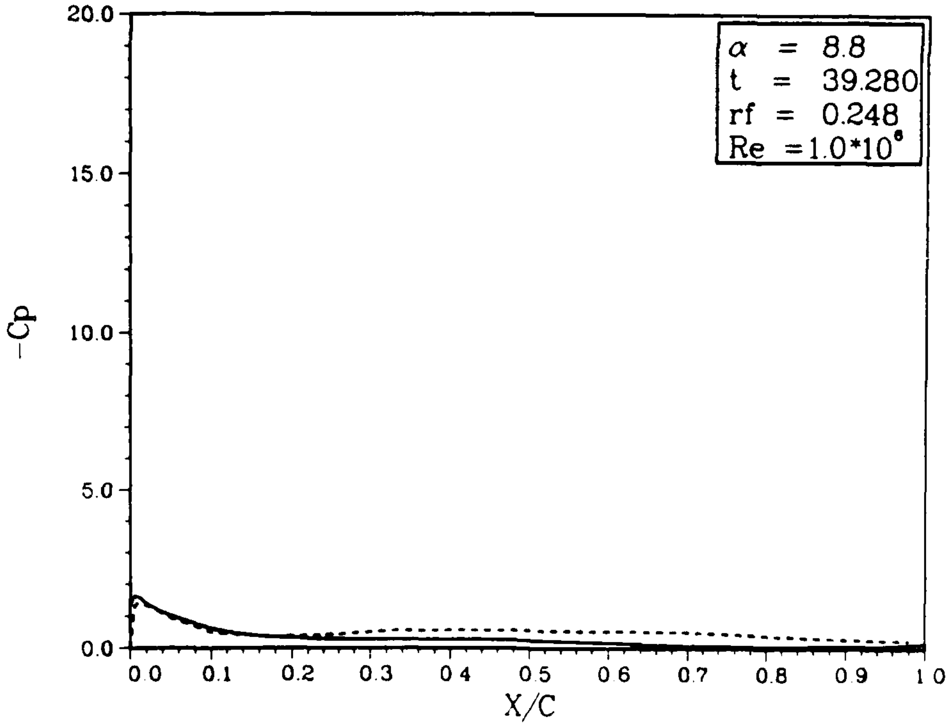

α = 22.9
t = 27.760
rf = 0.248
Re =1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

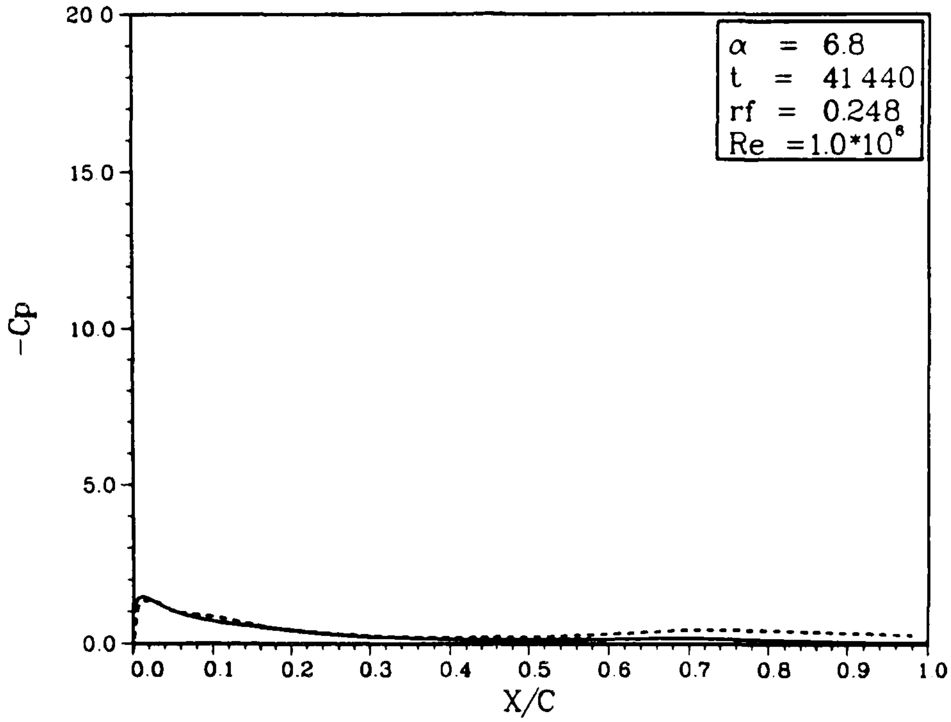

α = 20.4
t = 30.240
rf = 0.248
Re =1.0*10^6
-Cp
X/C
20.0
15.0
10.0
5.0
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

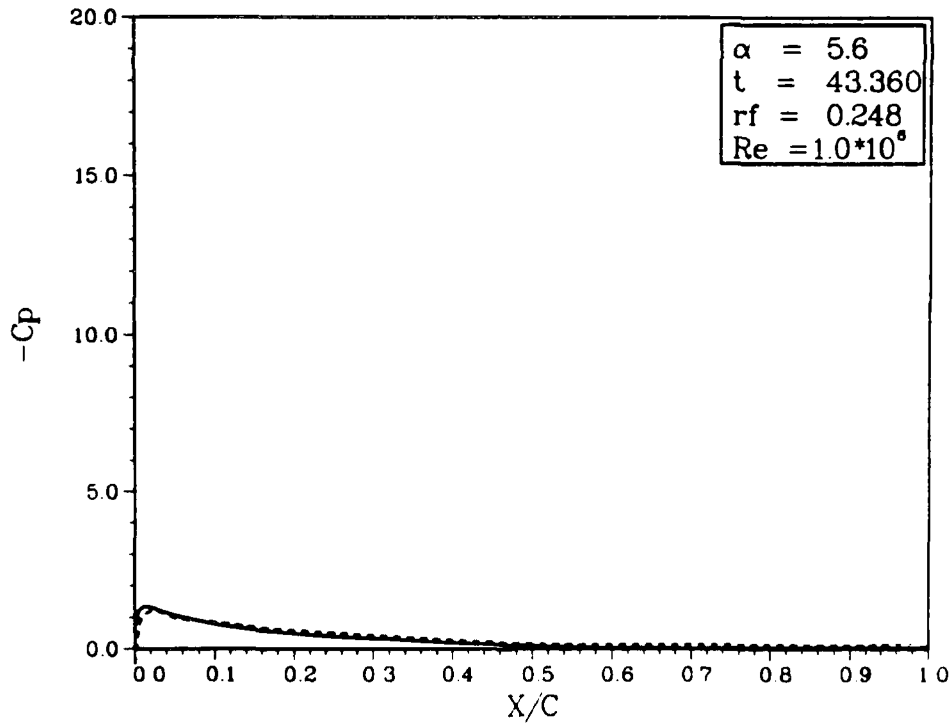

20.0
15.0
10.0
5.0
0.0
-Cp
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C
α = 17.6
t = 32.480
rf = 0.248
Re = 1.0*10^6

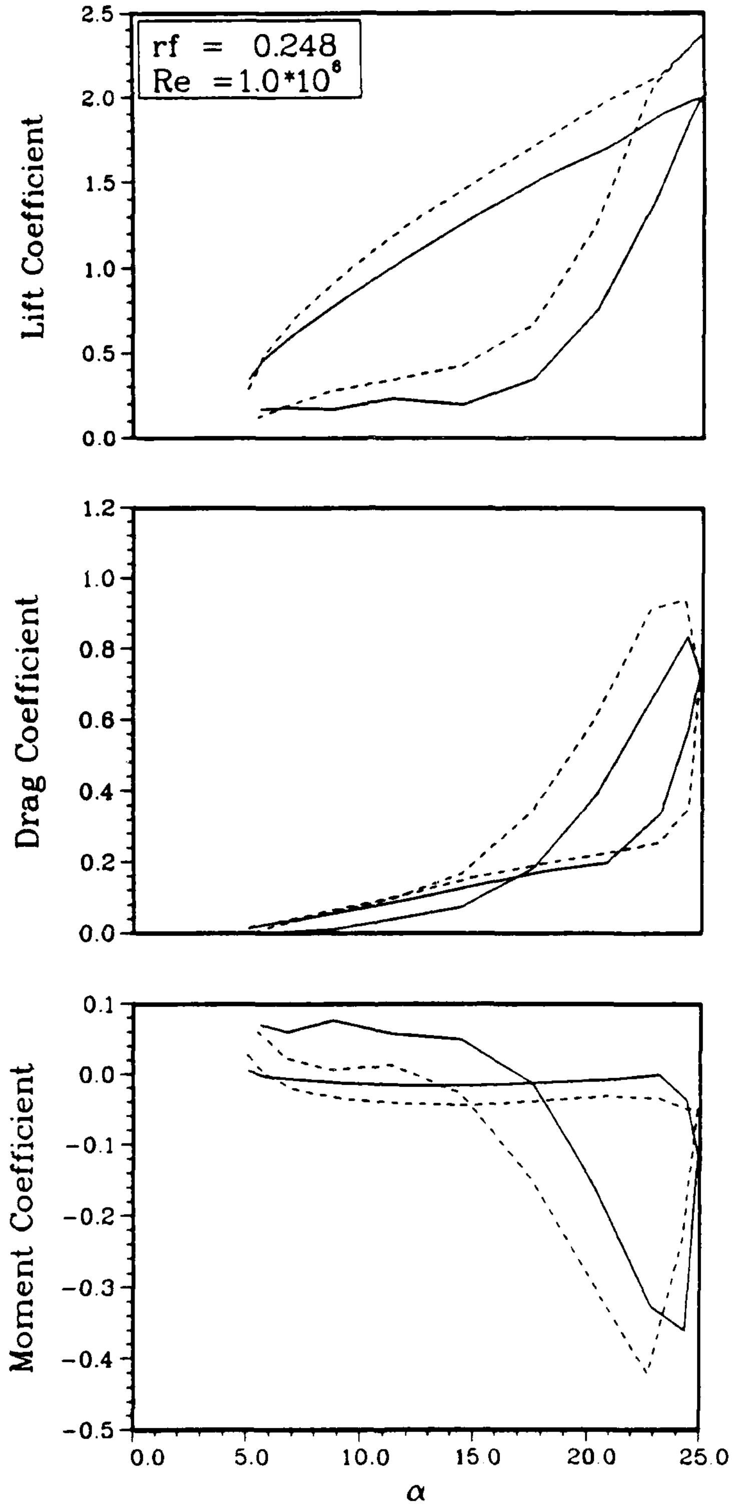

20.0
15.0
10.0
5.0
0.0
-Cp
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C
α = 14.4
t = 34.800
rf = 0.248
Re = 1.0*10^6

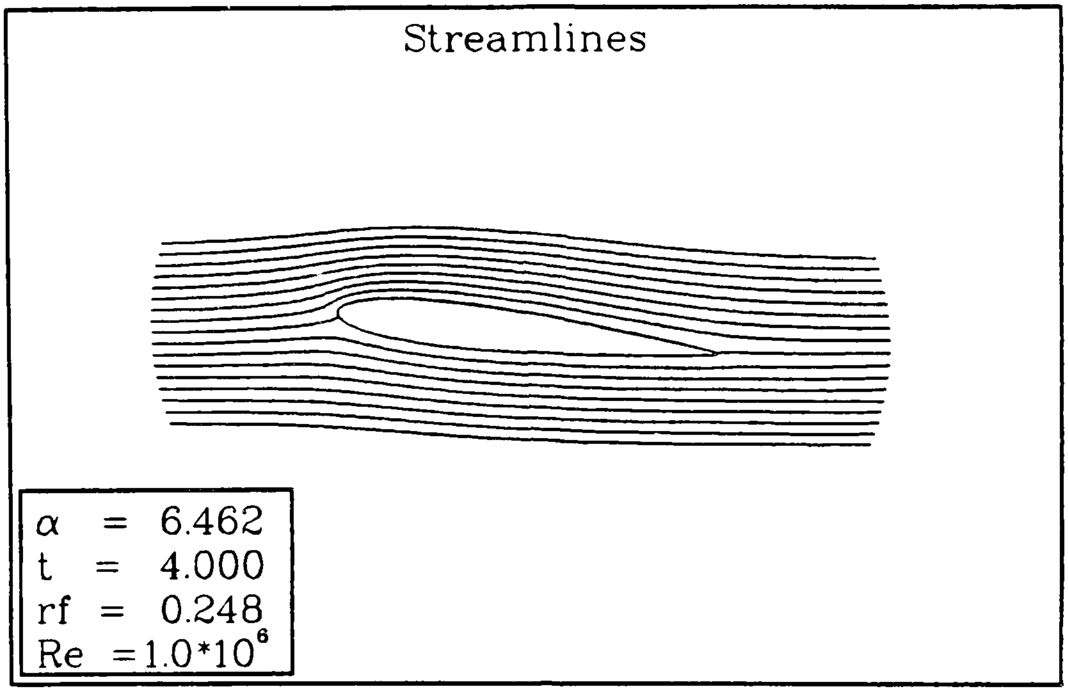

α = 11.3
t = 37.120
rf = 0.248
Re = 1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

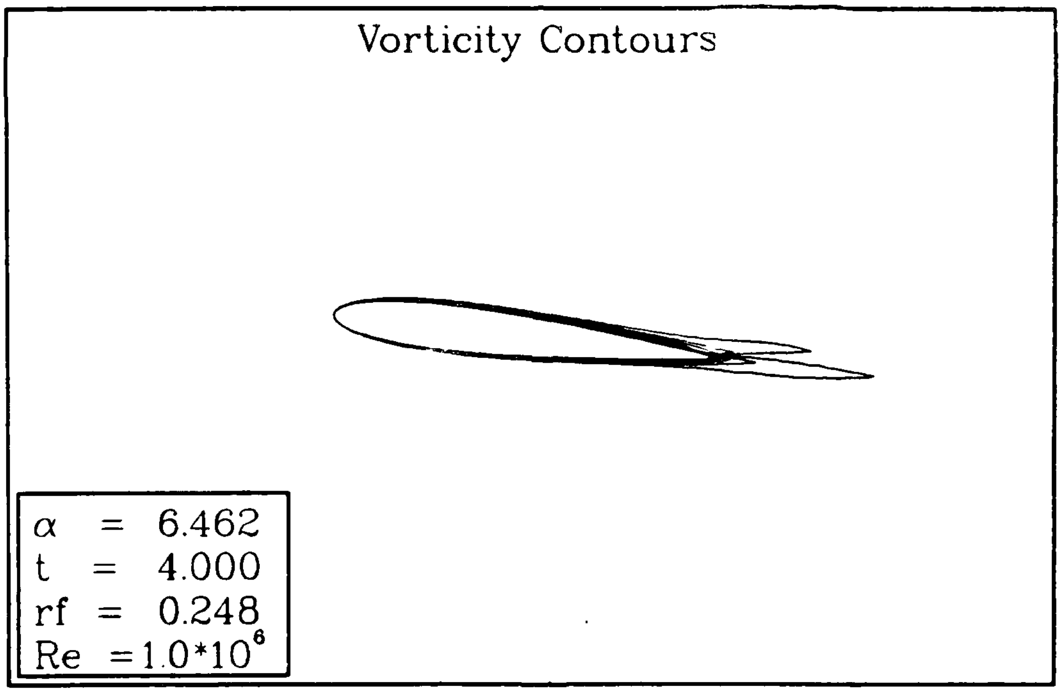

α = 8.8
t = 39.280
rf = 0.248
Re = 1.0*10^6
20.0
15.0
10.0
5.0
0.0
-Cp
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
X/C

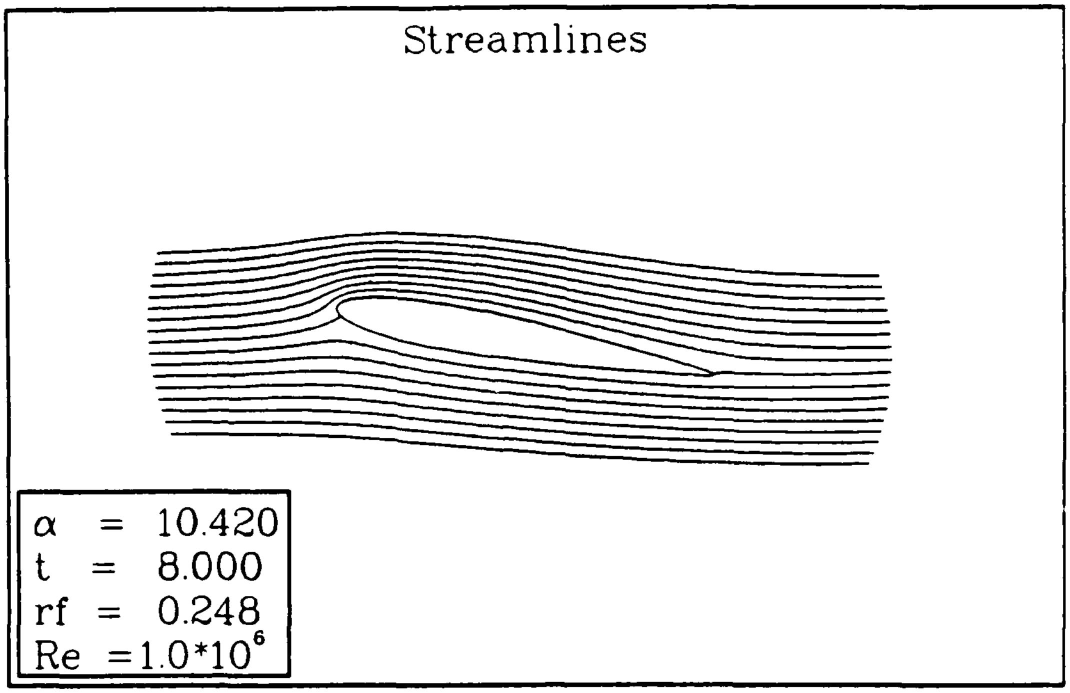

α = 6.8
t = 41.440
rf = 0.248
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C

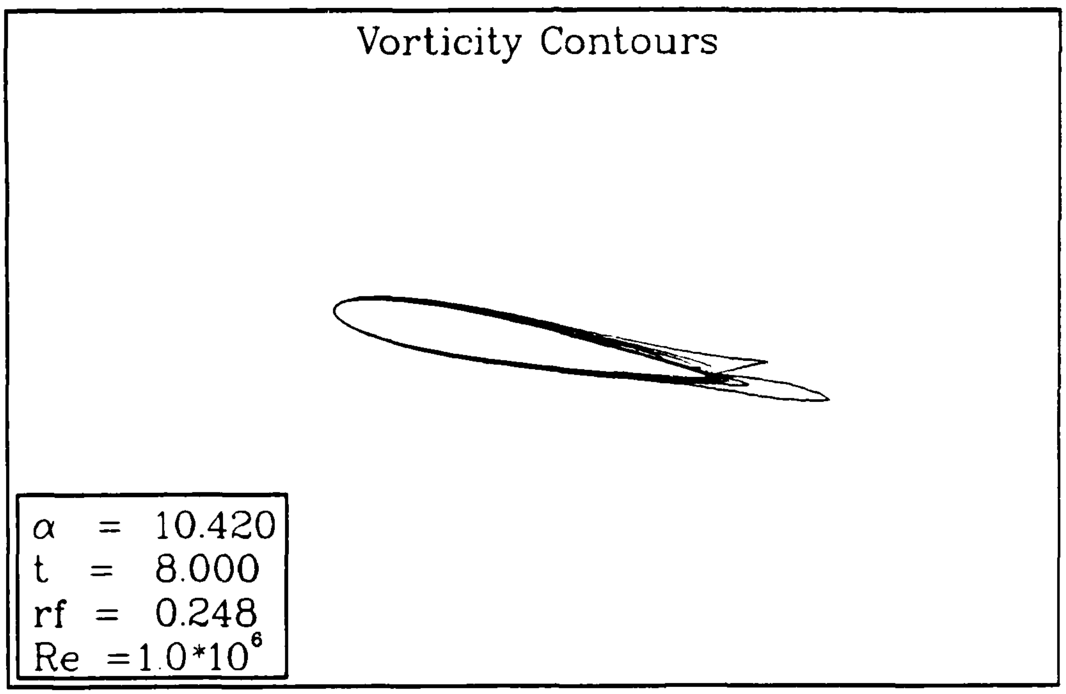

α = 5.6
t = 43.360
rf = 0.248
Re =1.0*10^6
-Cp
20.0
15.0
10.0
5.0
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
X/C

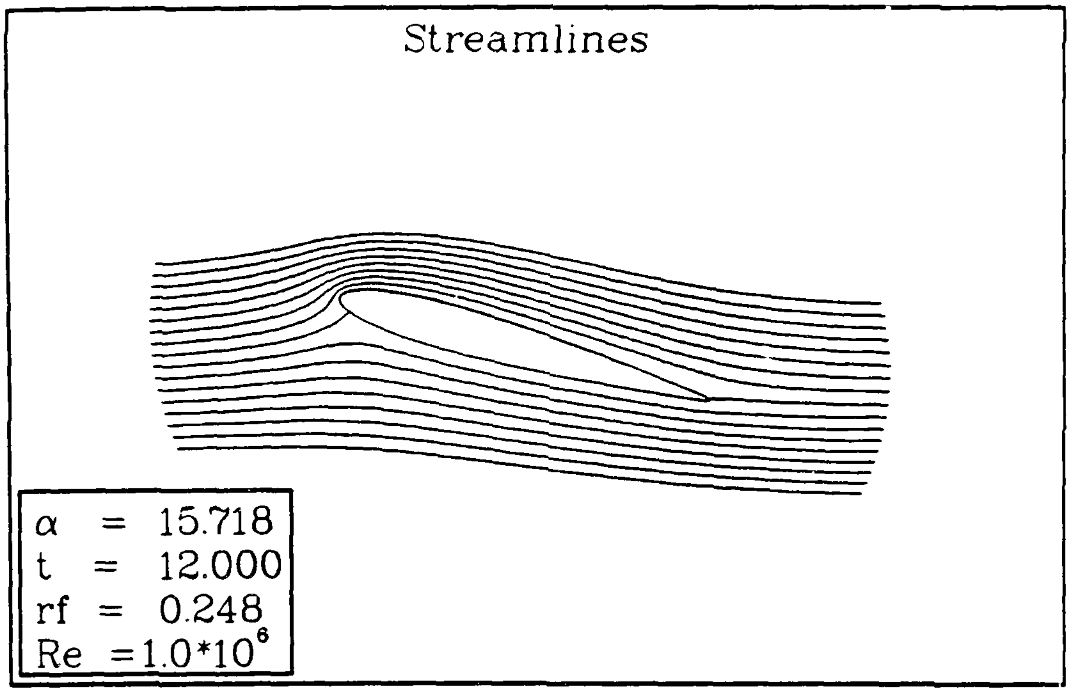

rf = 0.248
Re =1.0*10⁶
Lift Coefficient
Drag Coefficient
Moment Coefficient
0.0
5.0
10.0
15.0
20.0
25.0
α

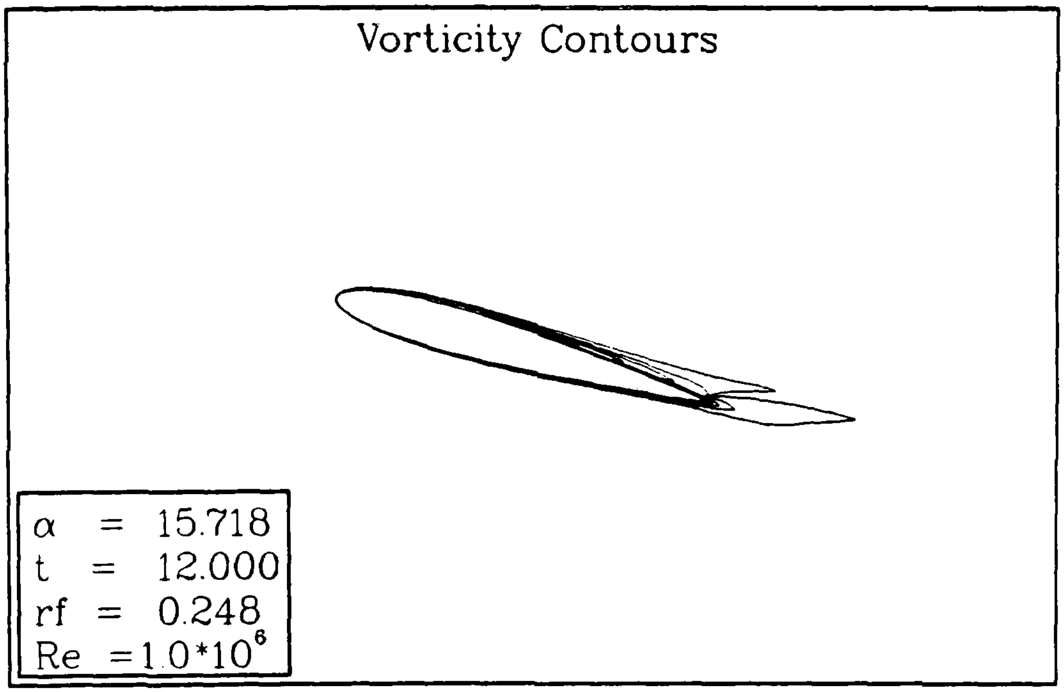

Streamlines
α = 6.462
t = 4.000
rf = 0.248
Re =1.0*10^6

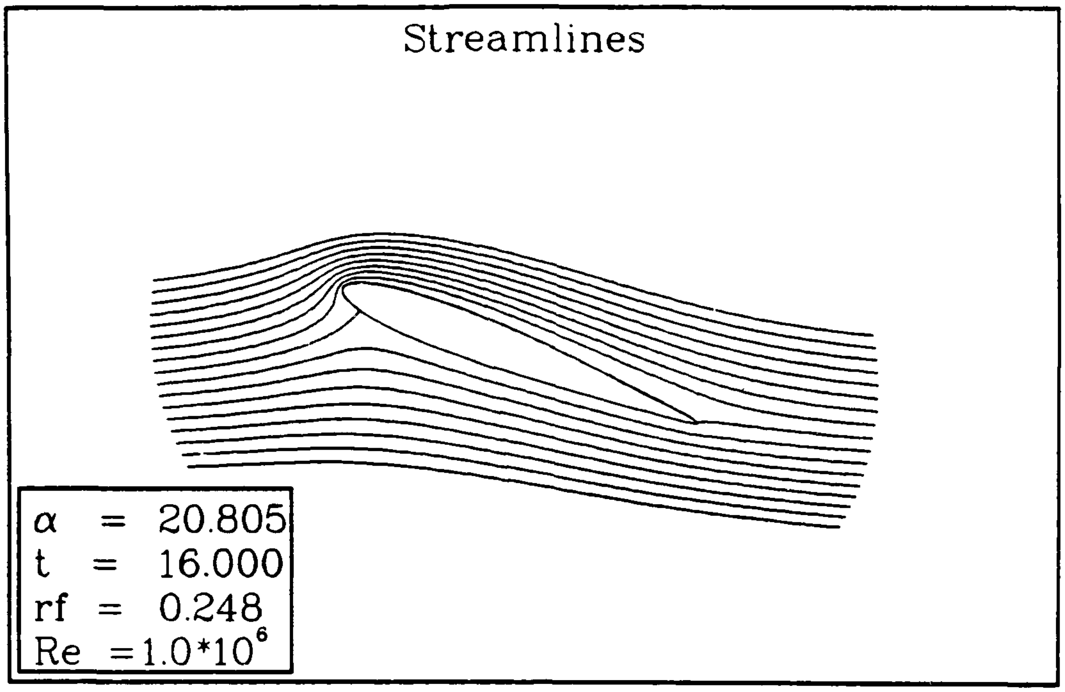

Vorticity Contours
α = 6.462
t = 4.000
rf = 0.248
Re =1.0*10^6

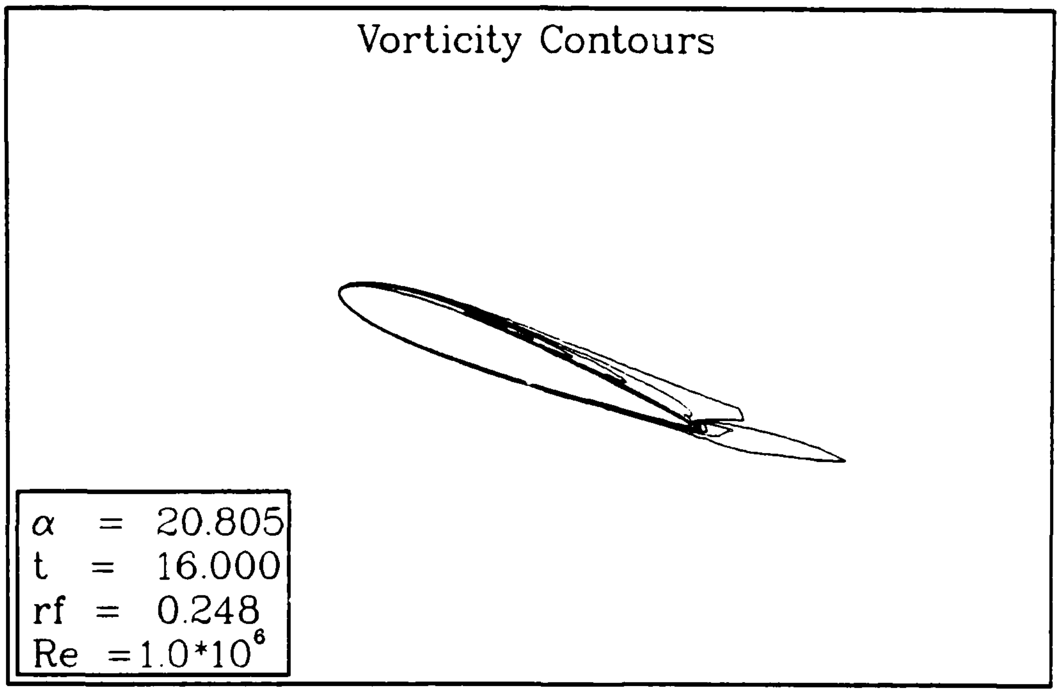

Streamlines
α = 10.420
t = 8.000
rf = 0.248
Re =1.0*10^6

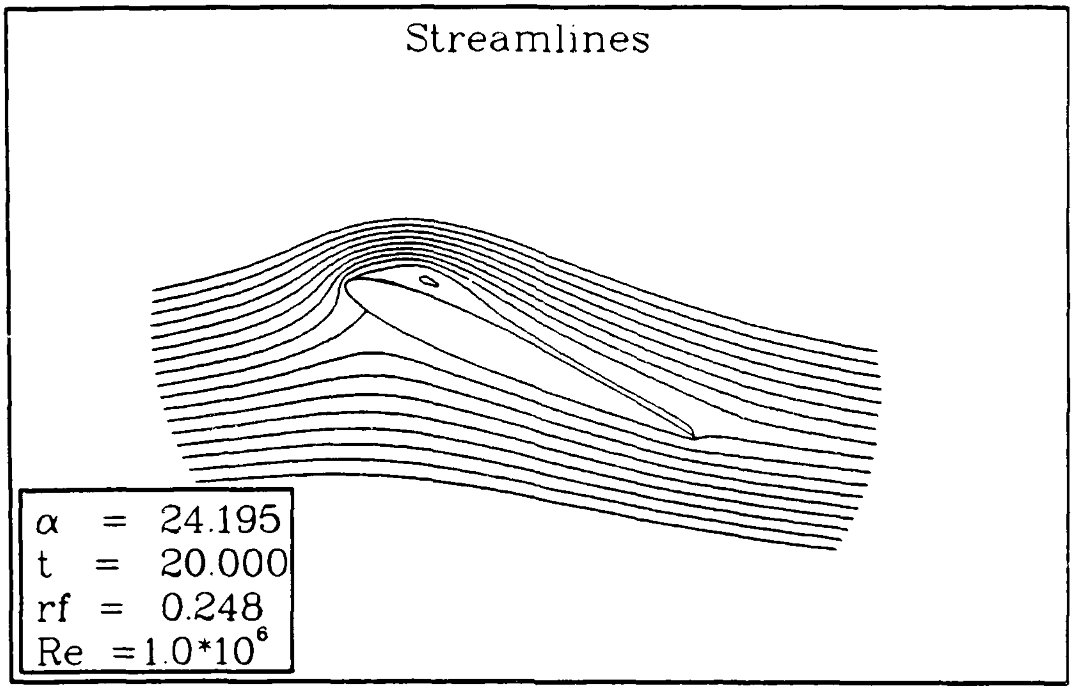

Vorticity Contours
α = 10.420
t = 8.000
rf = 0.248
Re =1.0*10^6

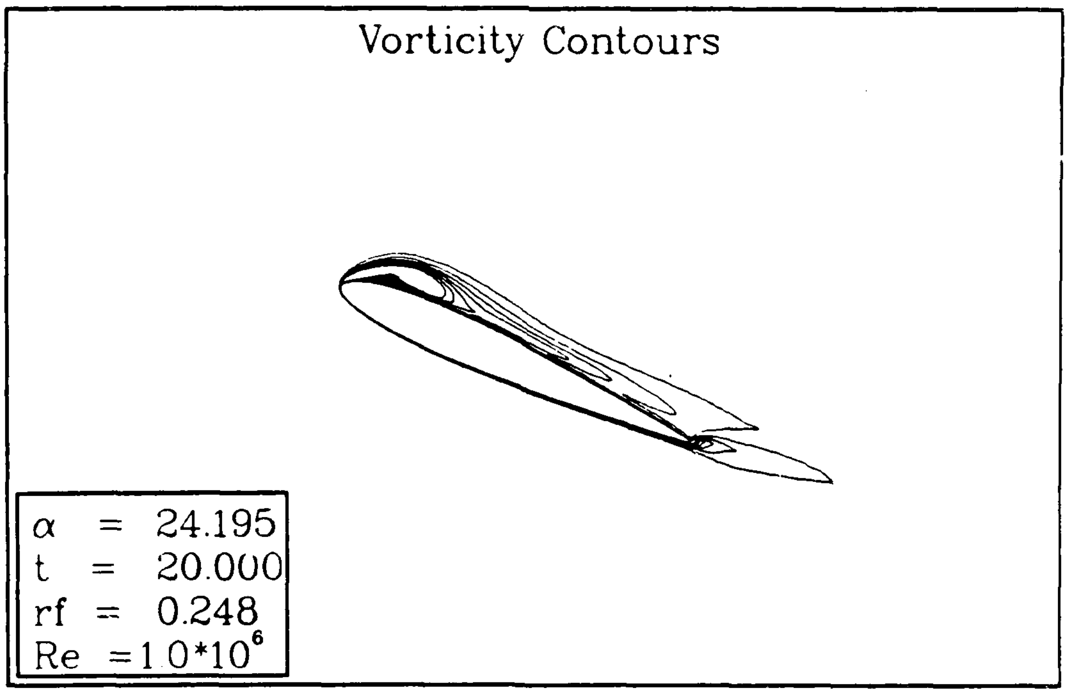

Streamlines
α = 15.718
t = 12.000
rf = 0.248
Re =1.0*10^6

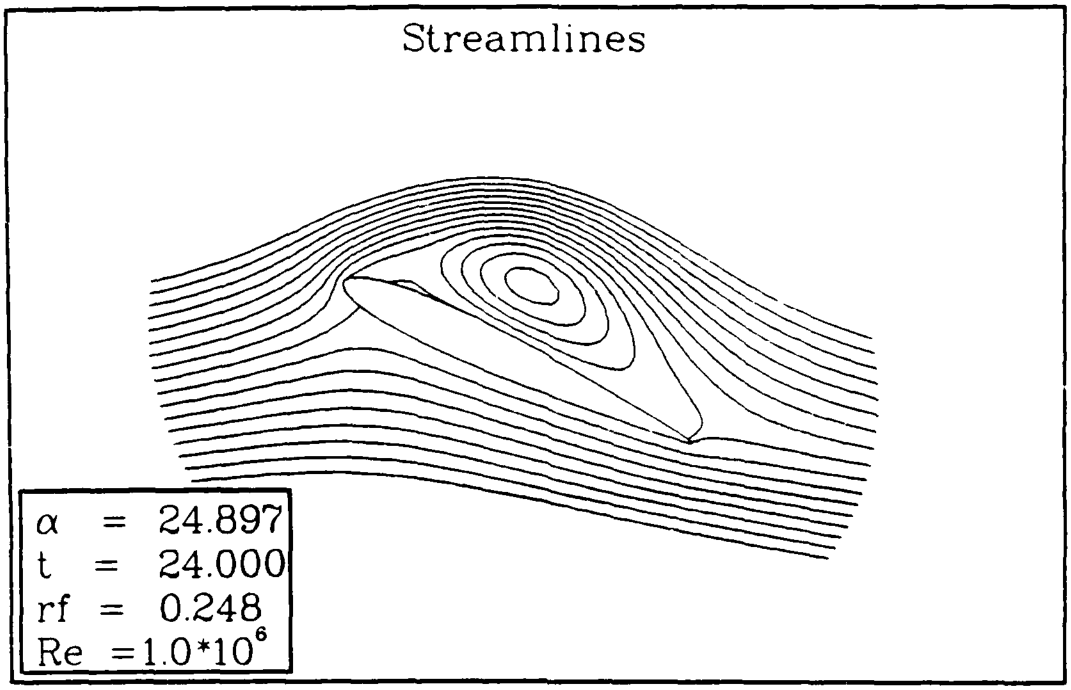

Vorticity Contours
α = 15.718
t = 12.000
rf = 0.248
Re =1.0*10^6

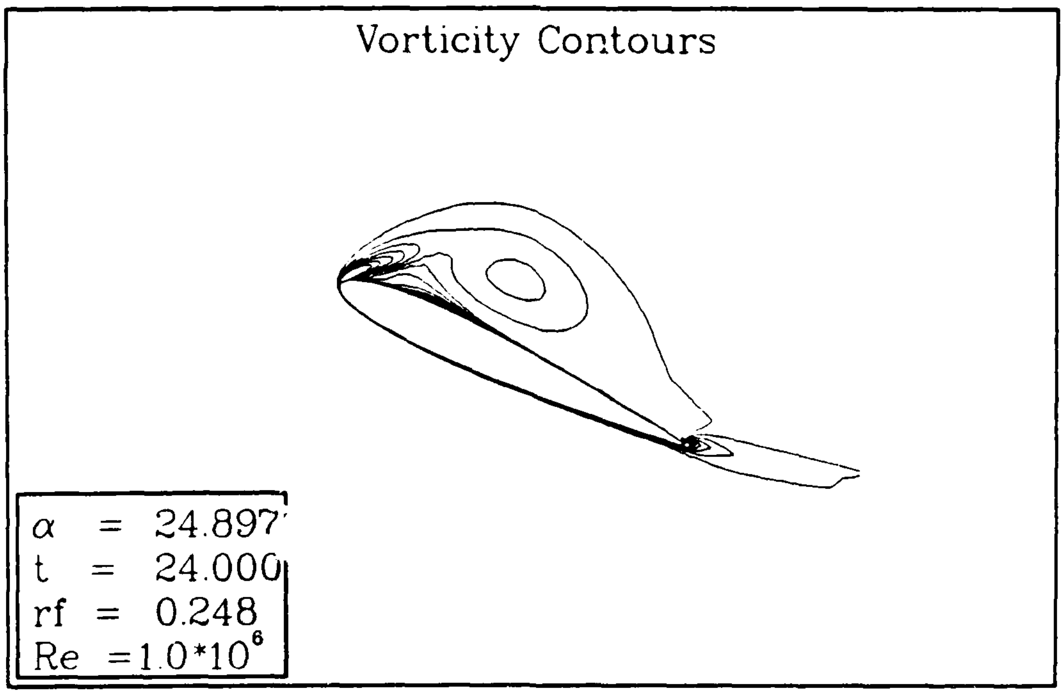

Streamlines
α = 20.805
t = 16.000
rf = 0.248
Re =1.0*10^6

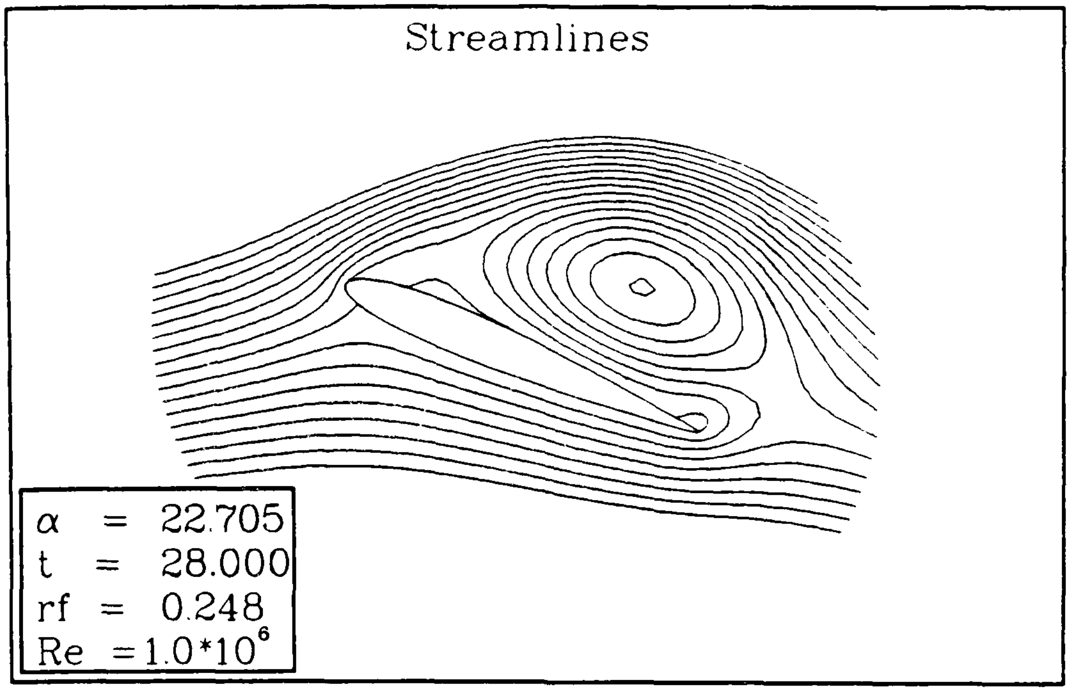

Vorticity Contours
α = 20.805
t = 16.000
rf = 0.248
Re =1.0*10^6

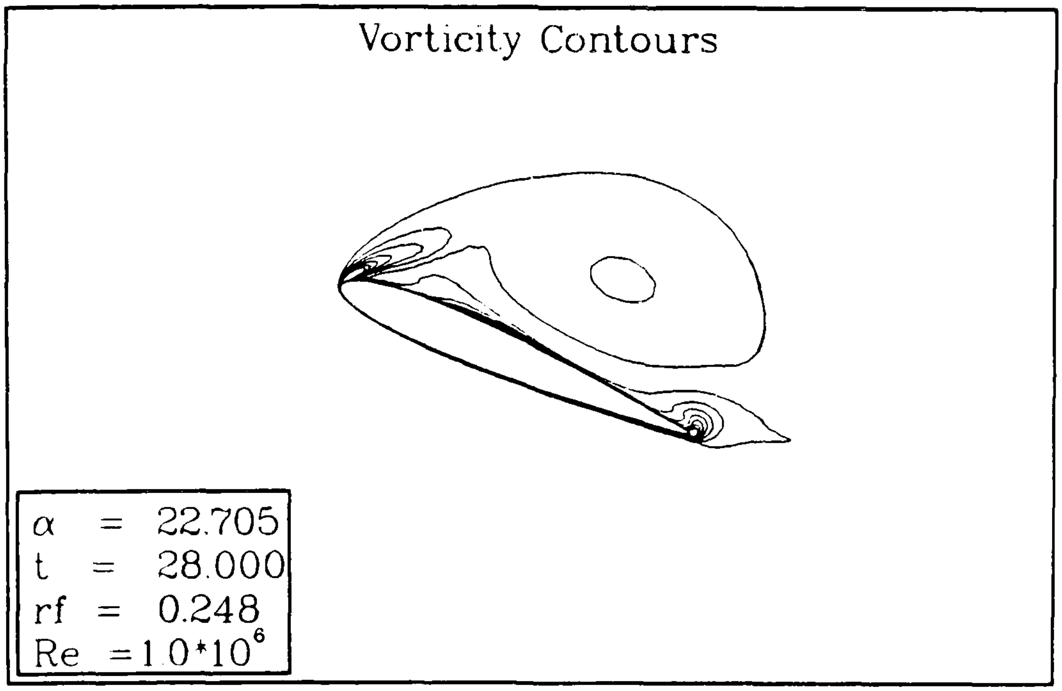

Streamlines
α = 24.195
t = 20.000
rf = 0.248
Re =1.0*10^6

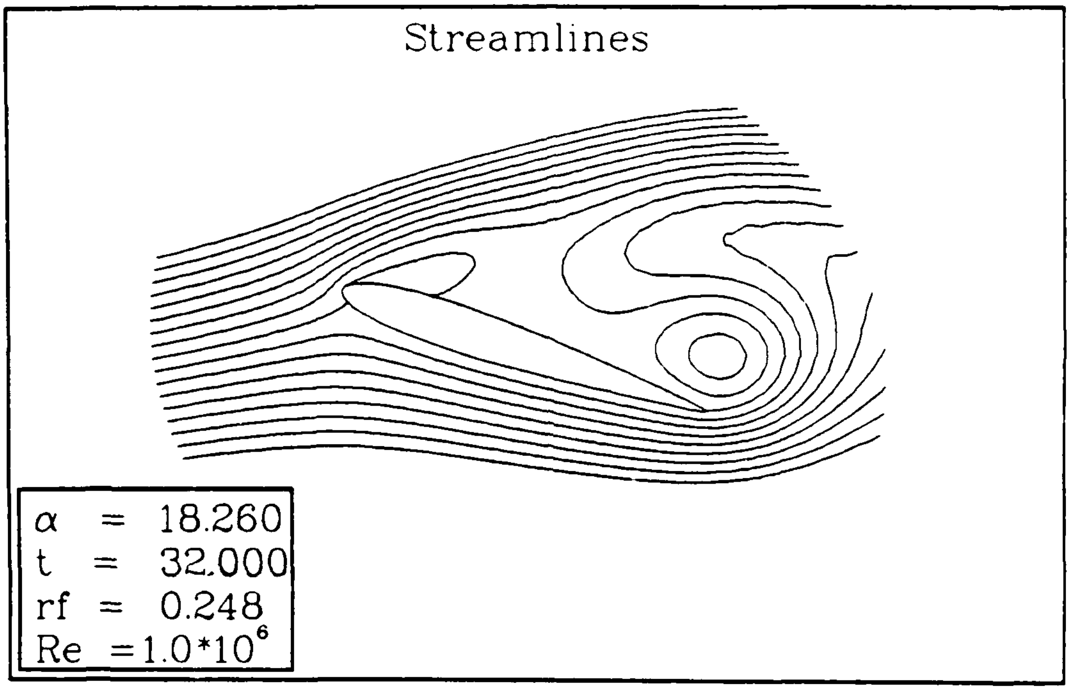

Vorticity Contours
α = 24.195
t = 20.000
rf = 0.248
Re =1.0*10^6

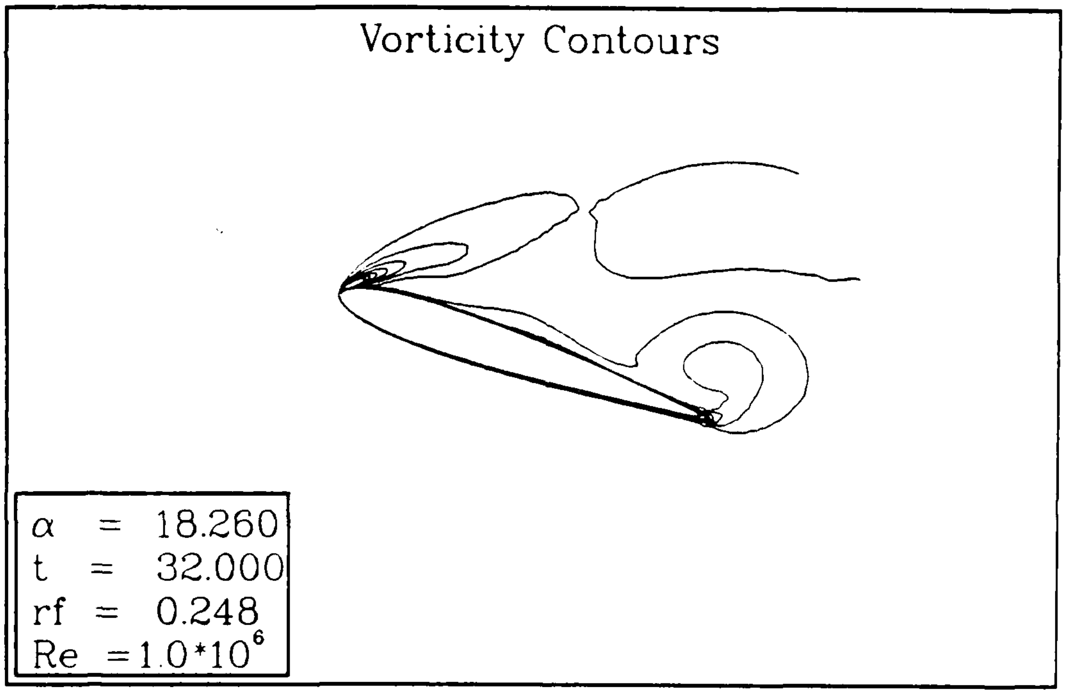

Streamlines
α = 24.897
t = 24.000
rf = 0.248
Re =1.0*10^6

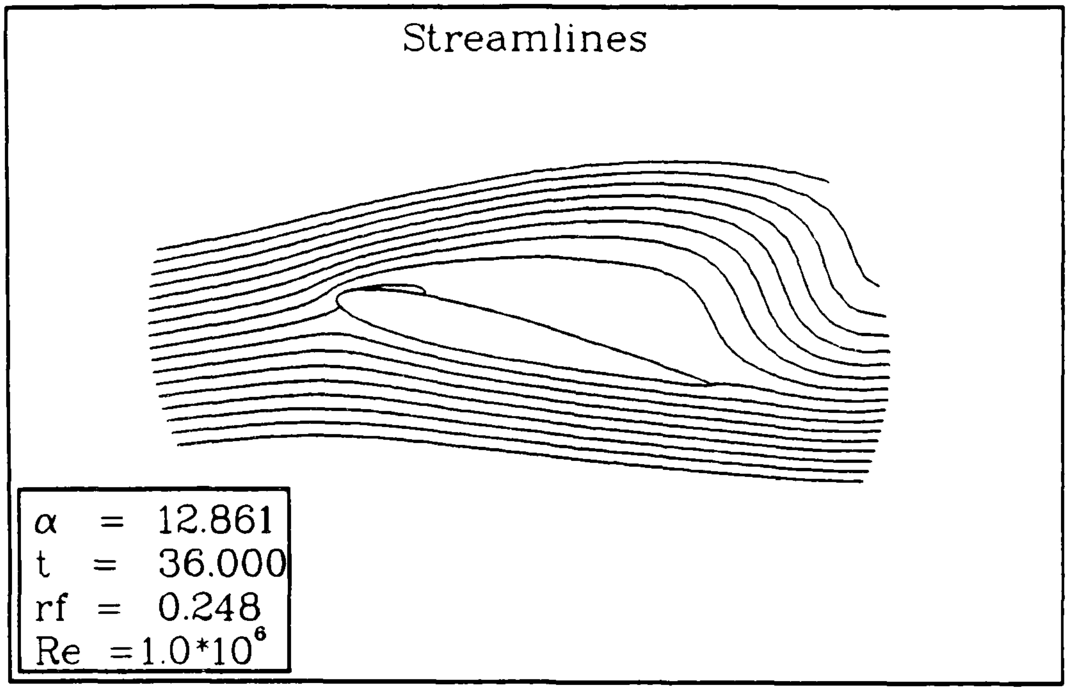

Vorticity Contours
α = 24.897
t = 24.000
rf = 0.248
Re =1.0*10^6

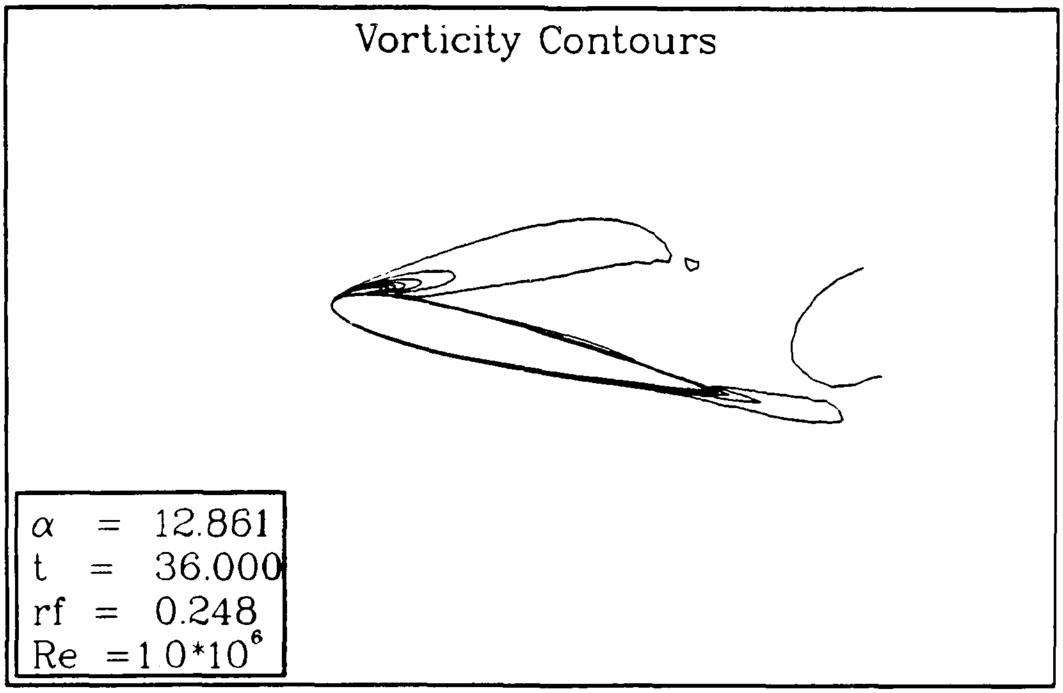

Streamlines
α = 22.705
t = 28.000
rf = 0.248
Re =1.0*10^6

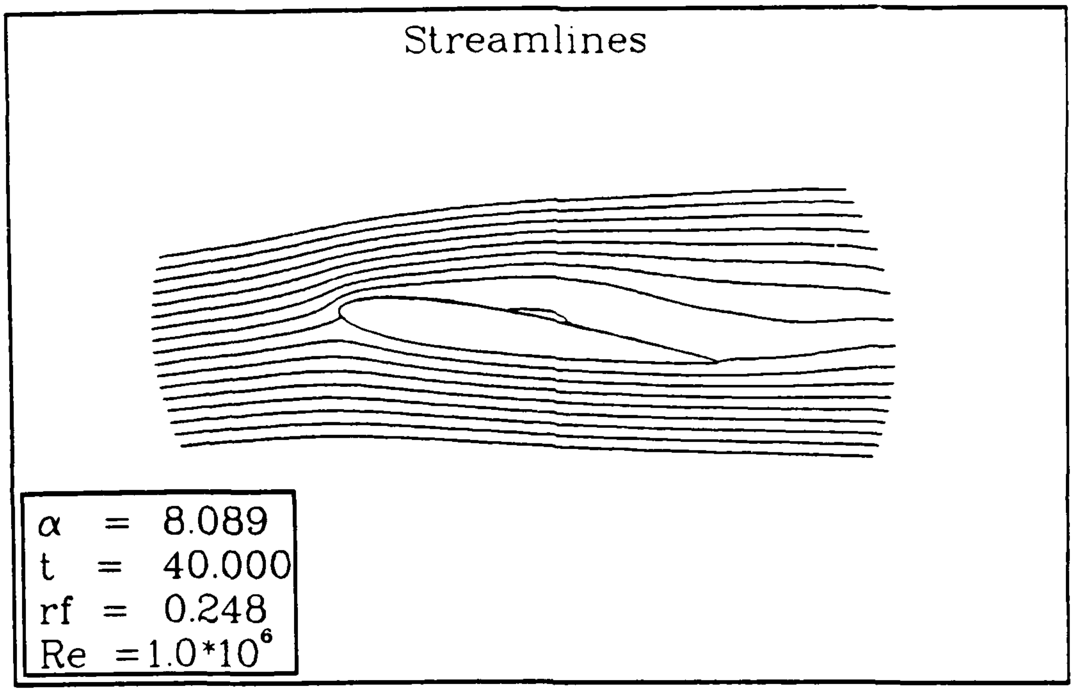

Vorticity Contours
α = 22.705
t = 28.000
rf = 0.248
Re =1.0*10^6

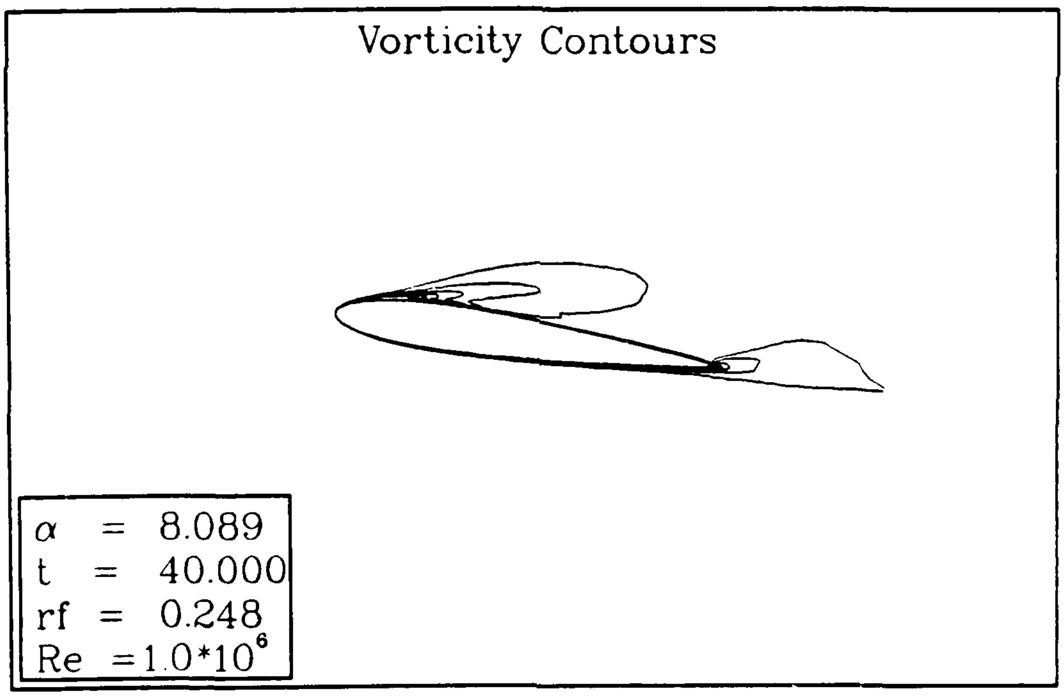

Streamlines
α = 18.260
t = 32.000
rf = 0.248
Re =1.0*10^6

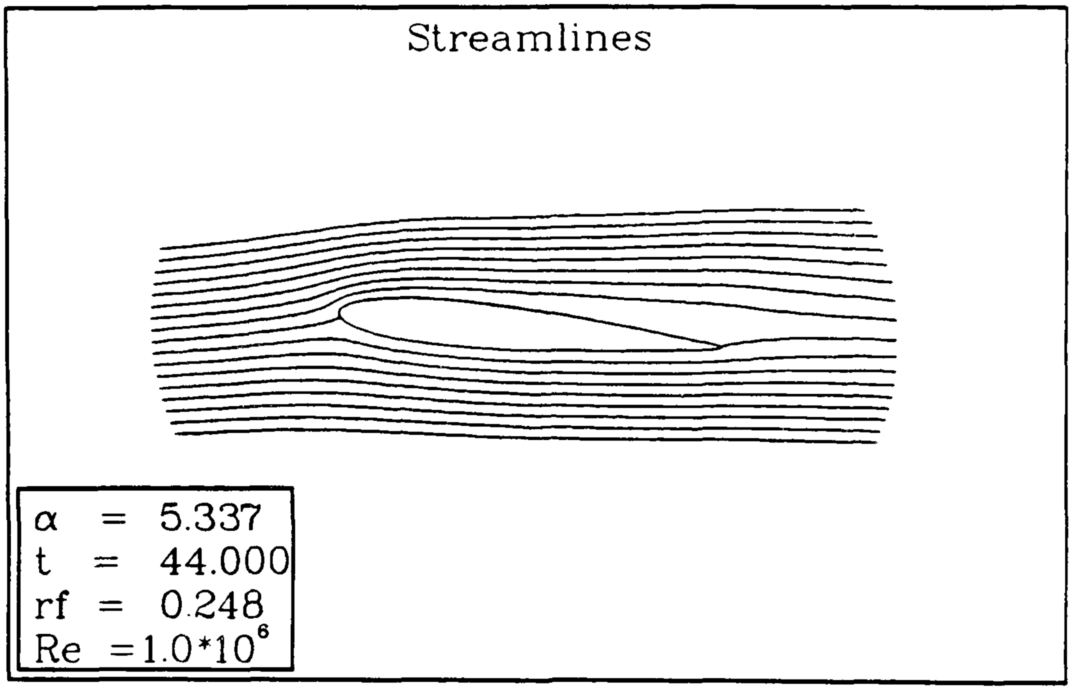

Vorticity Contours
α = 18.260
t = 32.000
rf = 0.248
Re =1.0*10^6

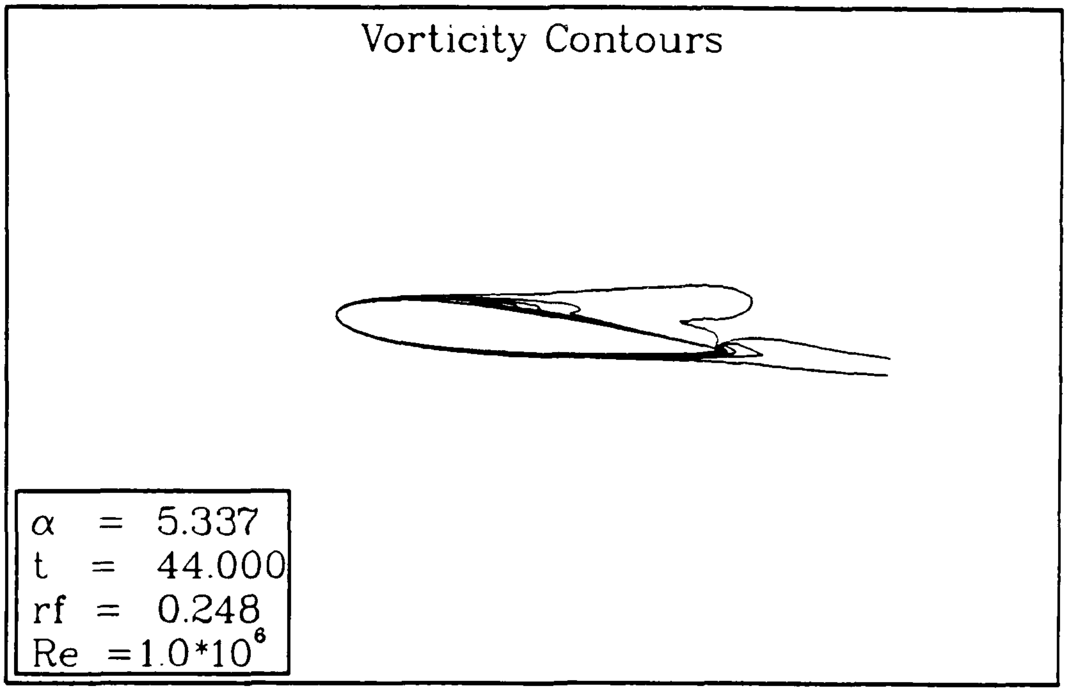

Streamlines
α = 12.861
t = 36.000
rf = 0.248
Re =1.0*10^6


Vorticity Contours
α = 12.861
t = 36.000
rf = 0.248
Re =1 0*10^6


Streamlines
α = 8.089
t = 40.000
rf = 0.248
Re =1.0*10^6


Vorticity Contours
α = 8.089
t = 40.000
rf = 0.248
Re =1.0*10^6


Streamlines
α = 5.337
t = 44.000
rf = 0.248
Re =1.0*10^6


Vorticity Contours
α = 5.337
t = 44.000
rf = 0.248
Re =1.0*10^6

APPENDIX B

CODE LISTING

```
      PROGRAM GEOM

C  **************************************************
C   GEOMETRY PLOTTING PROGRAM - DISSPLA VERSION
C   LAST REVISION 4-1-87
C
C   PRINCIPAL INVESTIGATOR @D DR. J.C. WU
C   AUTHORS @D MIKE PATTERSON, ISHMAEL TUNCER
C             GEORGIA INSTITUTE OF TECHNOLOGY
C             (404) 894-3028
C
C   TAPE1   @D INPUT TO PLOT1
C   TAPE2   @D input TO ZONST
C   TAPE3   @D INPUT TO LOADS
C   TAPE5   @D GENERAL INPUT
C   TAPE6   @D GENERAL OUTPUT
C   TAPE10  @D XYZ INPUT TO PLOT3D
C   TAPE14  @D INPUT TO PLOT2, PLOT3
C
C   CALLS   @D NONE
C  **************************************************

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41,KFC2=42)
      PARAMETER (INOR1=20)

      DIMENSION UC(IP1,JP1),VC(IP1,JP1),HREAL(IDIM),HIMAG(JDIM)
      DIMENSION CS(KFC1,IDIM),SN(KFC1,IDIM),HSTAR(IDIM,JDIM)
      DIMENSION R1(JP1),R2(JP1),R1D(JP1),RP(JP1,KFC2),RL(JP1)
      DIMENSION H(IDIM,JDIM),SGMA(KFC1)
      DIMENSION A2(JDIM),A4(JDIM),C2(JDIM),C4(JDIM),D5(JDIM)
      DIMENSION AS2(KFC1),BS2(KFC1),CS2(KFC1),DS2(KFC1)
      DIMENSION YN(KFC1,JDIM),CCP(KFC1,JDIM)
      DIMENSION IWK(JDIM,JDIM),INOR(INOR1,JDIM),COEF(IDIM,2)
      DIMENSION X(IDIM,JP1),Y(IDIM,JP1),XB(IDIM),YB(IDIM)
      DIMENSION XP(IP1,JP1),YP(IP1,JP1),XT(IDIM),YT(IDIM)
      COMPLEX W,W1,Z,Z1,DZDW,DWDT,HIAMG,HSTAR

      READ(5,*) RE
      READ(5,*) C1,CC2,DS
      READ(5,*) CSQ,GAMA,SIGMA,AL

      IM=IDIM
      JR=JDIM
      JR1=JP1
      KFC=KFC1
      PI=4.*ATAN(1.)
      DTET=2.*PI/FLOAT(IM)
      IM2=IM/2

C..COMPUTE RADIAL GRID DISTRIBUTION IN COMPUTATIONAL PLANE

      DO 100 J=1,JR1
        S=FLOAT(J-1)*DS
```

```
        R1(J)=EXP(S+C1)+CC2
        R2(J)=EXP(S+C1+.5*DS)+CC2
  100 CONTINUE
      DO 105 J=1,JR
  105 R1D(J)=R1(J+1)-R1(J)

C..COMPUTE SCALE FACTOR OF TRANSFORMATION

      DO 110 J=1,JR
      DO 110 I=1,IM
        PHI=FLOAT(I-1)*DTET
        WR=R2(J)*COS(PHI)
        WI=R2(J)*SIN(PHI)
        W=CMPLX(WR,WI)
        Z1=W+GAMA
        DZDW=1.-CSQ/(Z1*Z1)
        H1=CABS(DZDW)
        H(I,J)=H1*H1
        HREAL(I) = REAL(DZDW)
        HIMAG(J) = AIMAG(DZDW)
        HSTAR(I,J) = CMPLX(HREAL(I),-HIMAG(J))
  110 CONTINUE
      WRITE(20) HSTAR

C..ARRAYS USED IN KINETICS

      DO 115 J=1,JR
        A2(J)=R1(J+1)/(DS*(R2(J)-CC2))
        A4(J)=-A2(J)*AL/(RE*DS*(R1(J+1)-CC2))
        C2(J)=-R1(J)/(DS*(R2(J)-CC2))
        C4(J)=C2(J)*AL/(RE*DS*(R1(J)-CC2))
        D5(J)=AL/(R2(J)*RE*DTET*DTET)
  115 CONTINUE

C..COMPUTE GEOMETRIC COEFFICIENTS FOR COMPUTATION OF VELOCITY
C  FOURIER COEFFICIENTS BY INTEGRAL RELATIONS

      DO 120 J=1,JR1
        RP(J,1)=R1(J)
        RL(J)=LOG(R1(J))
        DO 120 K=2,KFC+1
        RP(J,K)=RP(J,K-1)*RP(J,1)
  120 CONTINUE

C..COMPUTE COORDINATES AND DERIVATIVES OF SOLID SURFACE
C  FOR USE IN LOADS CALCULATIONS

      DO 125 I=1,IM
        PHI=FLOAT(I-1)*DTET
        CO=COS(PHI)
        SI=SIN(PHI)
        W=CMPLX(CO,SI)
        DWDT=CMPLX(-SI,CO)
        Z1=W+GAMA
        Z=Z1+CSQ/Z1+SIGMA
        XB(I)=REAL(Z)
        YB(I)=AIMAG(Z)
        DZDW=1.-CSQ/(Z1*Z1)
        XT(I)=REAL(DZDW*DWDT)
        YT(I)=AIMAG(DZDW*DWDT)
  125 CONTINUE

C..COMPUTE CORRELATIONS OF VELOCITY COMPONENTS BETWEEN
C  INERTIA COORDINATE SYSTEM AND BODY COORDINATE SYSTEM
```

```
      DO 130 J=1,JR1
      DO 130 I=1,IM
        PHI=FLOAT(I-1)*DTET
        WR=R1(J)*COS(PHI)
        WI=R1(J)*SIN(PHI)
        W1=WR+GAMA
        W2=W1*W1+WI*WI
        XR=W1+CSQ*W1/W2+SIGMA
        YR=WI-CSQ*WI/W2
        XC=1.+CSQ*(WI*WI-W1*W1)/(W2*W2)
        XN=-CSQ*2.*WI*W1/(W2*W2)
        UC(I,J)=XR*XN+YR*XC
        VC(I,J)=YR*XN-XR*XC
  130 CONTINUE
      DO 135 J=1,JR1
        UC(IM+1,J)=UC(1,J)
        VC(IM+1,J)=VC(1,J)
  135 CONTINUE

C..COMPUTE COORDINATES OF VORTICITY GRID IN PHYSICAL PLANE
C  AND IN COMPUTATIONAL PLANE

      DO 140 J=1,JR1
      DO 140 I=1,IM
        PHI=FLOAT(I-1)*DTET
        WR=R2(J)*COS(PHI)
        WI=R2(J)*SIN(PHI)
        W=CMPLX(WR,WI)
        Z=W+GAMA+CSQ/(W+GAMA)+SIGMA
        XP(I,J)=WR
        YP(I,J)=WI
        X(I,J)=REAL(Z)
        Y(I,J)=AIMAG(Z)
  140 CONTINUE
        DO 141 J=1,JR1
        XP(IM+1,J)=XP(1,J)
        YP(IM+1,J)=YP(1,J)
  141 CONTINUE
      WRITE(10) IDIM,JDIM
      WRITE(10) ((X(I,J),I=1,IDIM),J=1,JDIM),
     1          ((Y(I,J),I=1,IDIM),J=1,JDIM)

      DO 145 I=1,IM
        PHI=FLOAT(I-1)*DTET
        CS(1,I)=1.
        SN(1,I)=0.
        CS(2,I)=COS(PHI)
        SN(2,I)=SIN(PHI)
  145 CONTINUE
      DO 150 K=3,KFC
      DO 150 I=1,IM
        L=1+MOD((K-1)*(I-1),IM)
        CS(K,I)=CS(2,L)
        SN(K,I)=SN(2,L)
  150 CONTINUE

C..COMPUTE FOURIER SERIES SMOOTHING FUNCTION

      DO 155 K=2,KFC
        A1=FLOAT(K-1)*DTET
        SGMA(K)=SIN(A1)/A1
  155 CONTINUE
```

```
C..DETERMINE FOURIER COEFFS. OF BOUNDARY VORTICITY CONDITION
C  ON INERTIA COOR. (COEF(I,1) FOR RADIAL COMP., COEF(I,2) FOR
C  TANGENTIAL COMP.)

      DO 160 K=1,KFC
        AS2(K)=0.
        BS2(K)=0.
        CS2(K)=0.
        DS2(K)=0.
  160 CONTINUE
      DO 161 I=1,IM
        COEF(I,1)=-(UC(I,1)*CS(2,I)+VC(I,1)*SN(2,I)) / FLOAT(IM2)
        COEF(I,2)=-(VC(I,1)*CS(2,I)-UC(I,1)*SN(2,I)) / FLOAT(IM2)
  161 CONTINUE
      DO 162 I=1,IM
        AS2(1)=AS2(1)+COEF(I,1)
        CS2(1)=CS2(1)+COEF(I,2)
        DO 162 K=2,KFC
          AS2(K)=AS2(K)+COEF(I,1)*CS(K,I)
          BS2(K)=BS2(K)+COEF(I,1)*SN(K,I)
          CS2(K)=CS2(K)+COEF(I,2)*CS(K,I)
          DS2(K)=DS2(K)+COEF(I,2)*SN(K,I)
  162 CONTINUE
      DO 163 K=2,KFC
        AS2(K)=AS2(K)*SGMA(K)
        BS2(K)=BS2(K)*SGMA(K)
        CS2(K)=CS2(K)*SGMA(K)
        DS2(K)=DS2(K)*SGMA(K)
  163 CONTINUE

C..COMPUTE 'YN'

      DO 165 I=1,KFC
      DO 165 J=1,JR
      IS=I
      IF(I .GT. IM/3) THEN
        YN(I,J)=SQRT((X(I,J)-XB(I))**2+(Y(I,J)-YB(I))**2)
      ELSE
        DIF=X(I,J)-XB(I)
        IF(ABS(DIF) .GT. 0.001) THEN
          INC=1
          IF(DIF .GT. 0.) INC=-1
  164     IS=IS+INC
          IF(IS .GT. KFC .OR. IS .LT. 1) THEN
             YN(I,J)=Y(I,J)
          ELSE
             DIFN=X(I,J)-XB(IS)
             IF(DIFN*DIF .LT. 0.) THEN
                ISM=IS-INC
                YIS=YB(ISM)+(X(I,J)-XB(ISM))*(YB(IS)-YB(ISM))
     1              /(XB(IS)-XB(ISM))
                YN(I,J)=Y(I,J)-YIS
             ELSE
                GO TO 164
             ENDIF
          ENDIF
        ELSE
          YN(I,J)=Y(I,J)-YB(I)
        ENDIF
      ENDIF
  165 CONTINUE

C..COMPUTE 'INOR'
```

```
      DO 170 II=2,IM/4
      I=II
      INOR(II,1)=I
      DO 170 J=2,JR
        INC=1
        IF(X(I,J) .LT. X(II,1)) INC=-1
        IF(ABS(X(II,1)-X(I+INC,J)) .LT. ABS(X(II,1)-X(I,J))) I=I+INC
        INOR(II,J)=I
  170 CONTINUE

C..COMPUTE 'IWK'

      DO 175 JN=1,JR-1
      I=INOR(2,JN)
      IWK(JN,JN)=I
      DO 175 J=JN+1,JR
        INC=1
        IF(Y(I,J) .GT. Y(I,JN)) INC=-1
        IF(ABS(Y(I,JN)-Y(I+INC,J)) .LT. ABS(Y(I,JN)-Y(I,J))) I=I+INC
        IWK(JN,J)=I
  175 CONTINUE

C..CALCULATE GEOMETRIC COEFF. ON PRESSURE COMP.

      DO 180 J=1,JR
        DO 179 I=3,KFC
  179   CCP(I,J)= 1. / FLOAT(I-2)*(1./R1(J)**(I-2)
     1            -1./R1(J+1)**(I-2))
        CCP(1,J)= R1D(J)
        CCP(2,J)= LOG(R1(J+1)/R1(J))
  180 CONTINUE

C..INTEGRATE OVER AIRFOIL SURFACE TO COMPUTE AIRFOIL AREA;
C  ASSUMES AIRFOIL IS SYMMETRIC

      DPHI=PI/200.
      CO=-1.
      SI=0.
      W=CMPLX(CO,SI)
      Z1=W+GAMA
      Z=Z1+CSQ/Z1
      X1=REAL(Z)
      Y1=AIMAG(Z)
      PHI=PI
      A1=0.
      DO 200 I=1,200
        PHI=PHI-DPHI
        CO=COS(PHI)
        SI=SIN(PHI)
        W=CMPLX(CO,SI)
        Z1=W+GAMA
        Z=Z1+CSQ/Z1
        X2=REAL(Z)
        Y2=AIMAG(Z)
        A1=A1+.5*(Y2+Y1)*(X2-X1)
        X1=X2
        Y1=Y2
  200 CONTINUE
      AF=2.*A1

C..OUTPUT TO THE VARIOUS FILES

      WRITE(1) AL,X,Y,INOR,IWK
```

```
      WRITE(2) DTET,DS,AF,AL,RE
      WRITE(2) CS,SN,UC,VC,H
      WRITE(2) RP,RL,SGMA
      WRITE(2) A2,A4,C2,C4,D5
      WRITE(2) AS2,BS2,CS2,DS2
      WRITE(2) R1,R2,R1D,YN
      WRITE(2) INOR,IWK,CCP

      WRITE(3) AL,RE,DTET,XB,YB,XT,YT
      WRITE(3) CSQ,GAMA,SIGMA,C1,CC2

      WRITE(6,10)
      WRITE(6,40) IM,JR,SIGMA,CSQ,C1,CC2,DS,GAMA,AL,RE
      WRITE(6,20) (H(I,1),I=1,IM)
      WRITE(6,30) R1

      WRITE(14) CSQ,GAMA,SIGMA,AL
      WRITE(14) XB,YB,XP,YP

   10 FORMAT(//,'INPUT TO GEOM')
   20 FORMAT(//,'SQUARE OF THE SCALE FACTORS AT FIRST RING@D'
     1 ,//(10E13.6))
   30 FORMAT(//,'    GRID DISTRIBUTION IN R DIRECTION @D '//(10E13.6))
   40 FORMAT(//,/1X,
     1  'IM= ',T10,I5,/1X,
     2  'JR= ',T10,I5,/1X,
     3  'SIGMA= ',T10,F10.7,/1X,
     4  'CSQ= ',T10,F10.7,/1X,
     5  'C1= ',T10,F10.7,/1X,
     6  'CC2= ',T10,F10.7,/1X,
     7  'DS= ',T10,F10.7,/1X,
     8  'GAMA= ',T10,F10.7,/1X,
     9  'AL= ',T10,F10.7,/1X,
     1  'RE= ',T10,F10.1)

      STOP
      END
```

```
1000000
-4 927978847591   9927588760838 .1293603008237
.8061367   -.0647384  .9170954997604  3.619058974007
```

```
INPUTS:     NACA 0012   REVISED PARAMETERS

RE
C1,C2,DS
CSQ,GAMA,SIGMA,AL
```

```
      PROGRAM ZONST
C  *******************************************************
C    2-D INCOMPRESSIBLE NAVIER-STOKES SOLVER - DISSPLA VERSION
C    LAST REVISION@D  5-20-87
C
C    PRINCIPAL INVESTIGATOR @D DR. J.C. WU
C    AUTHORS @D MIKE PATTERSON, ISHMAEL TUNCER
C               GEORGIA INSTITUTE OF TECHNOLOGY
C               (404) 894-3028
C
C    TAPE2   @D OUTPUT FROM GEOM
C    TAPE4   @D OUTPUT FOR LOADS
C    TAPE5   @D GENERAL INPUT
C    TAPE6   @D GENERAL OUTPUT
C    TAPE7   @D INPUT FROM PREVIOUS RUN
C    TAPE8   @D OUTPUT FOR NEXT RUN
C    TAPE9   @D OUTPUT FOR PLOT2
C    TAPE11  @D Q INPUT FOR PLOT3D
C
C    CALLS   @D KMTCS
C               KNTCS
C               CPVAL
C  *******************************************************

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41,KFC2=42)
      PARAMETER (IDUM=4295,INOR1=20)

      DIMENSION AS2(KFC1),BS2(KFC1),CS2(KFC1),DS2(KFC1)
      COMMON/IO/LOOP,NT,NTMAX,NTOUT
      COMMON/DKIN/A2(JDIM),A4(JDIM),C2(JDIM),C4(JDIM),D5(JDIM)
      COMMON/VLB/VORLB,NLB
      COMMON/RGRD/R1D(JP1),R2(JP1)
      COMMON/SMT/SGMA(KFC1)
      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/RHS/WOW(IP1,JP1),POP(IP1,JP1),WP(IP1),WB(IP1),R1(JP1),
     1        DUMM(IDUM)
      COMMON/ABCDS/AS1(KFC1),BS1(KFC1),CS1(KFC1),DS1(KFC1)
      COMMON/COR/RP(JP1,KFC2),RL(JP1)
      COMMON/DELTA/DS,DTET,DT
      COMMON/UNF/URR,URF,DFMX,NCC
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/TRIG/CS(KFC1,IDIM),SN(KFC1,IDIM)
      COMMON/SCALE/H(IDIM,JDIM)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/VEL/U(IDIM,JP1),V(IDIM,JP1),HSTAR(IDIM,JDIM)
      COMMON/TUR1/USTAR(IDIM),EDDY(IDIM,JDIM),IOTUR(IDIM),
     1         YN(KFC1,JDIM),IOT,IOTB
      COMMON/TUR2/INOR(INOR1,JDIM),IWK(JDIM,JDIM)
      COMMON/VEH/UC(IP1,JP1),VC(IP1,JP1)
      COMMON/ZNS/IB1,IV1,IV2,IB2,IV1R
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/CPC/CCP(KFC1,JDIM),CP(IP1)
      COMMON Q1(IDIM,JDIM),Q2(IDIM,JDIM),Q3(IDIM,JDIM),Q4(IDIM,JDIM)
      COMPLEX HSTAR

      REAL ALLP(2)
      DATA ALLP/ 19.5,20.0/
```

```
C      REWIND 2
C      REWIND 4
C      REWIND 5
C      REWIND 6
C      REWIND 7
C      REWIND 8
C      REWIND 9
       REWIND 20

C..READ GENERAL INPUTS AND ECHO THEM

      READ(5,*) ICST,RF,ALPS,ICTUR
      READ(5,*) WMIN,DFMX,DRMX,KMAX,NCC
      READ(5,*) URBI,URBP,URR
      READ(5,*) IV1,IB1,IB2,IV2,NPL,NLB
      READ(5,*) DTI,DTINC,NTMAX
      READ(5,*) NTPL,NTOUT,NTLO

      WRITE(6,54)
      WRITE(6,*) ICST,RF,ALPS,ICTUR
      WRITE(6,*) WMIN,DFMX,DRMX,KMAX,NCC
      WRITE(6,*) URBI,URBP,URR
      WRITE(6,*) IV1,IB1,IB2,IV2,NPL,NLB
      WRITE(6,*) DTI,DTINC,NTMAX
      WRITE(6,*) NTPL,NTOUT,NTLO

C..INPUTS FROM GEOM

      READ(2) DTET,DS,AF,AL,RE
      READ(2) CS,SN,UC,VC,H
      READ(2) RP,RL,SGMA
      READ(2) A2,A4,C2,C4,D5
      READ(2) AS2,BS2,CS2,DS2
      READ(2) R1,R2,R1D,YN
      READ(2) INOR,IWK,CCP
      READ(20) HSTAR

      NALLP=1
      IM=IDIM
      JR=JDIM
      KFC=KFC1
      URF=1.-URR
      NLBP=NLB+1
      PI=4.*ATAN(1.)
      IM1=IM+1
      IV1R=IM+IV1
      IM2=IM/2
      RDTET=DTET*R1(NLBP)
      VSC=AL/RE
      FR=2.*RF/AL
      ALP=ALPS*PI/180.
      OMG=0.
      OMGD=0.

C..START INITIAL SOLUTION OR READ PREVIOUS ITERATION RESULTS

      IF(ICST.LT.2) THEN
        DO 100 K=1,KFC
          AS1(K)=0.
          BS1(K)=0.
          CS1(K)=0.
          DS1(K)=0.
  100   CONTINUE
```

```
      ENDIF
      IF(ICST.EQ.0) THEN
        NT=0
        IF(DTI.EQ.0.) DT=0.08
        T=0.0
        N=15
        VORLB=0.0
        UI=COS(ALP)
        VI=SIN(ALP)
        DO 105 I=1,IM
          KIN(I)=N
          U(I,1)=0.
          V(I,1)=0.
          DO 105 J=1,JR
          VOR(I,J)=0.
  105   CONTINUE
        AA(1,1)=0.
        AA(1,2)=2.*VI/(RP(2,1)-RP(1,1))
        BB(1,2)=-2.*UI/(RP(2,1)-RP(1,1))

C..POTENTIAL FLOW SOLUTION

        CALL KMTCS
        DO 106 I=1,IM
  106   VOR(I,1)=(AA(1,2)*CS(2,I)+BB(1,2)*SN(2,I))/H(I,1)
      ENDIF

C .READ PREVIOUS RUN AND RESET TIME IF A NEW MOTION IS SPECIFIED

      I=ICST
      IF(ICST.GE.1) READ(7) ICST,NT,N,KIN,T,DT,VOR,U,V,VORLB

      IF(I.GT.ICST) THEN
        IF(I.NE.1) THEN
        WRITE(6,51)
        NT=0
        T=0.0
        ENDIF
      ENDIF
      ICST=I
      IF(DTI.NE.0.) DT=DTI

C  //////////TIME STEP LOOP////////////////////
C .START COMPUTATIONS FOR SUBSEQUENT TIME STEPS
C  FROM TIME LEVEL NS1 TO NTMAX (DO LOOP 1001)

      DO 1001 LOOP=1,NTMAX
      TO=T
      NTO=NT
      DTO=DT
      DT=DT+DTINC
      IF (DT.GT..08) DT=.08
      T=T+DT
      NT=NT+1
      DO 110 J=1,N
      DO 110 I=1,IM
  110 VOROLD(I,J)=VOR(I,J)

C..DEFINE AIRFOIL MOTION

      IF(ICST.EQ.2) THEN
        ALP=(T-SIN(7.5*T)/7.5)*FR*0.5
        OMG=-(1.-COS(7.5*T))*FR*0.5
        OMGD=-SIN(7.5*T)*FR*0.5*7.5
```

```
      ENDIF
      IF(ICST.EQ.3) THEN
        ALP=FR*T
        OMG=-FR
        IF(ALP.GT.0.6) THEN
          ALP=0.6
          OMG=0.
        ENDIF
      ENDIF
      IF(ICST.EQ.4) THEN
        ALP=(15.-10.*COS(FR*T))*PI/180.
        OMG=-10.*FR*SIN(FR*T)*PI/180.
        OMGD=-10.*FR*FR*COS(FR*T)*PI/180.
      ENDIF

C..VELOCITY BOUNDARY CONDITION ON BODY

      IF(ICST.GT.1) THEN
        DO 115 K=1,KFC
          AS1(K)=AS2(K)*OMG
          BS1(K)=BS2(K)*OMG
          CS1(K)=CS2(K)*OMG
          DS1(K)=DS2(K)*OMG
  115   CONTINUE
      ENDIF
      ALPD=ALP*180./PI
      UI=COS(ALP)
      VI=SIN(ALP)
      IF (INT(ALPD*10.) .EQ. INT(ALLP(NALLP)*10.))THEN
         ICPL   = 0
         ICLD   = 0
         ICOUT  = 0
         NALLP  = NALLP+1
         WRITE(6,61) NT,T,DT,ALPD,OMG,OMGD
        ELSE
         ICPL   = 1
         ICLD   = 1
         ICOUT  = 1
        ENDIF

C       ICPL=MOD(NT,NTPL)
C       ICLD=MOD(NT,NTLO)
C       ICOUT=MOD(NT,NTOUT)
C       WRITE(6,61) NT,T,DT,ALPD,OMG,OMGD

C..SOLVER FOR KINETICS

      CALL KNTCS(WMIN,URBI,URBP,KMAX,DRMX,KODE,KKK,ALP)
      IF(KODE.EQ.1) GO TO 5000

C..UPDATE ARRAY KIN

      DO 120 I=1,IM
        JI=KEN(I)+3
        IF(JI.GT.JR) JI=JR
        IF(N.LT.JI) N=JI
        KIN(I)=JI
  120 CONTINUE
      IF(N.LT.NPL) N=NPL

C..SOLVER FOR KINEMATICS

      CALL KMTCS
```

```
C..EVALUATE VORTICITY LEAVING BOUNDARY

      DO 125 I=1,IM
      IF(KEN(I).GE.NLB) VORLB=VORLB+DT*RDTET*(V(I,NLBP)*0.5*(VOR(I,NLB)
     1 +VOR(I,NLBP))-VSC*(VOR(I,NLBP)-VOR(I,NLB))/(R2(NLBP)-R2(NLB)))
  125 CONTINUE

C..EVALUTE THE TOTAL VOPTICITY INCLUDING SOLID ROTATION

      WSUM1=2.*OMG*AF+VORLB
      DO 130 J=1,NLB
        ACA=R2(J)*DTET*R1D(J)
        DO 130 I=1,IM
        WSUM1=WSUM1+VOR(I,J)*H(I,J)*ACA
  130 CONTINUE
      WRITE(6,57) KKK,WSUM1,VORLB

C..CALCULATE CP

      IF(ICLD.EQ.0 .OR. ICPL.EQ.0 .OR. LOOP.EQ.NTMAX) THEN
        CALL CPVAL
        DO 135 I=1,IM
          IF(ICTUR.EQ.1 .AND. IOTUR(I).EQ.1) THEN
            WB(I)=USTAR(I)*USTAR(I)/VSC
            IF(VOR(I,1).LT.0.) WB(I)=-WB(I)
          ELSE
            WB(I)=VOR(I,1)
          ENDIF
  135   CONTINUE
        WB(IM1)=WB(1)
        WRITE(4) NT,NTPL,T,RF,WB,CP,OMG,OMGD,ALP
      ENDIF

C..OUTPUT AT EVERY NTOUT TIME STEP

      IF(ICOUT.EQ.0 .OR. LOOP.EQ.NTMAX) THEN
C        WRITE(6,63) (VOR(I,1),I=1,IM)
C        WRITE(6,67) (U(I,2),I=1,IM)
C        WRITE(6,69) (KEN(I),I=1,IM)
C        WRITE(21) W1,W2
C        IF(ICTUR.EQ.1) WRITE(6,70) IOT,IOTB,IM
      ENDIF

C..CALCULATE STREAM FUNCTION AND VORTICITY AT VELOCITY
C  GRID POINTS AND STORE ON TAPE FOR GENERATING PLOTS.
C  INTEGRATION IS FIRST ORDER TRAPAZOIDAL RULE.

      IF(ICPL.EQ.0) THEN
        DO 140 I=1,IM1
  140   POP(I,1)=0.

C..INTEGRATE TANGENTIAL VELOCITIES

        DO 150 I=1,IM
          DO 149 J=2,NPL
            WOW(I,J)=.5*(VOR(I,J)+VOR(I,J-1))
            POP(I,J)=POP(I,J-1)-.5*(U(I,J-1)+U(I,J))*R1D(J-1)
  149     CONTINUE
          WOW(I,1)=2.*VOR(I,1)-WOW(I,2)
  150   CONTINUE
        DO 151 J=1,NPL
          WOW(IM1,J)=WOW(1,J)
          POP(IM1,J)=POP(1,J)
  151   CONTINUE
```

```
          WRITE(9) NT,NPL,T,ALPD,RF,RE,WOW,POP
        ENDIF

 1001 CONTINUE
C  /////////////////END OF TIME STEP///////////////////////

C..ABNORMAL EXIT.  WRITE DATA FROM LAST COMPLETED TIME STEP

 5000 IF (KODE.EQ.1) THEN
          WRITE(6,52) KKK
          WRITE(6,53) NTO
          WRITE(6,63) (VOR(I,1),I=1,IM)
          WRITE(6,67) (U(I,2),I=1,IM)
          WRITE(6,69) (KEN(I),I=1,IM)
          IF(ICTUR.EQ.1) WRITE(6,70) IOT,IOTB,IM
          WRITE(8) ICST,NTO,N,KIN,TO,DTO,VOROLD,U,V,VORLB
          STOP
        ENDIF

C..NORMAL EXIT.  WRITE THE REQUIRED DATA FOR NEXT RUN

        WRITE(6,53) NT
        WRITE(8) ICST,NT,N,KIN,T,DT,VOR,U,V,VORLB
        WRITE(11) IDIM,JDIM
        WRITE(11) 1.,ALPD,1000000.,1.
        DO 600 I=1,IDIM
        DO 600 J=1,JDIM
        Q1(I,J)=1
  600   CONTINUE
        WRITE(11) ((Q1(I,J),I=1,IDIM),J=1,JDIM),
     1           ((Q2(I,J),I=1,IDIM),J=1,JDIM),
     2           ((Q3(I,J),I=1,IDIM),J=1,JDIM),
     3           ((Q4(I,J),I=1,IDIM),J=1,JDIM)
        STOP

   51 FORMAT(//20X,'*AIRFOIL MOTION HAS BEEN CHANGED, TIME RESET*',//)
   52 FORMAT(/10X,'*NO CONVER. SOLU. WITHIN',I5,3X,'ITERATIONS*',//)
   53 FORMAT(//,30X,'$$ NEXT RUN WITH NSTART =',I6,'  $$')
   54 FORMAT(//3X,'INPUT FILE TO ZONST'/)
   57 FORMAT(12X,'ITKNT =',I4,9X,'TVOR = ',E13.6,'    VORLB =',E13.6)
   61 FORMAT(/2X,90(1H-)/1X,' NT =',I4,',',',    T =',F8.4,'    DT =',F8.6
     1 ,'     AA =',F7.4,'     OMG =',F6.4,'     OMGD =',F6.4/2X,90(1H-))
   63 FORMAT(//1H ,'VORTICITY ---FIRST RING---'
     1 ,'CCW FROM TRAILING EDGE@D'/,/(10E13.6))
   67 FORMAT(/1H ,'TANGENTIAL VELOCITY ---J=2---'
     1 ,'CCW FROM TRAILING EDGE@D'/,/(10E13.6))
   69 FORMAT(/1H ,'VORTICITY EXTENT  @D       *** KEN '
     1 ,'***',/,/(1H ,4(2X,10I3)))
   70 FORMAT(/1H ,'TURBULENT REGION  @D   ',I3,'- 1    AND   ',I3,'-',I3)
        END
```

```
      SUBROUTINE KNTCS(WMIN,URBI,URBP,KMAX,DRMX,KODE,KKK,ALP)
C
C     ***********************************************************
C     KINETICS OF THE PROBLEM
C
C     CALLS      @D WSURFT   WSURF
C                   EDDYS    FOCFT
C                   VORTY
C
C     CALLED BY @D ZONST
C     ***********************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41)

      COMMON/IO/LOOP,NT,NTMAX,NTOUT
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/SGA/GA(IDIM,JDIM),GB(IDIM,JDIM),GC(IDIM,JDIM)
      COMMON/RHS/GD(IDIM,JDIM),DIP1(IDIM,JDIM),DIM1(IDIM,JDIM)
      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/TRIG/CS(KFC1,IDIM),SN(KFC1,IDIM)
      COMMON/DELTA/DS,DTET,DT
      COMMON/ZNS/IB1,IV1,IV2,IB2,IV1R
      COMMON/TUR1/USTAR(IDIM),EDDY(IDIM,JDIM),IOTUR(IDIM),
     1           YN(KFC1,JDIM),IOT,IOTB
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/SCALE/H(IDIM,JDIM)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/VEL/U(IDIM,JP1),V(IDIM,JP1)
      COMMON/RGRD/R1D(JP1),R2(JP1)
      COMMON/DKIN/A2(JDIM),A4(JDIM),C2(JDIM),C4(JDIM),D5(JDIM)
      COMMON/FM/GAM(IDIM,JDIM)

      KODE=0
      IKS=1
      IF(ALP.EQ.0.) IKS=IM2+2
      IF(MOD(NT,NTOUT).EQ.0 .OR. LOOP.EQ.NTMAX) WRITE(6,10)

C..COMPUTE FRICTION VELOCITY AND EDDY VISCOSITY

      IF(ICTUR.EQ.1) THEN
        CALL WSURFT
        CALL EDDYS
      ENDIF

C..CONSTRUCT DATA ARRAYS THAT ARE NEEDED IN COMPUTATION OF
C  FINITE DIFFERENCE FORM OF VORTICITY TRANSPORT EQUATION
C     TIME TERM         - FORWARD DIFFERENCE
C     DIFFUSION TERMS   - CENTRAL DIFFERENCES
C     CONVECTION TERMS - 2ND UPWIND DIFFERENCES

      DO 101 I=IKS,IM
      IPP1=I+1
      IM1=I-1
      IF(IPP1.GT.IM) IPP1=IPP1-IM
      IF(IM1.LT.1) IM1=IM+IM1
      JL=KIN(I)

C..DETERMINE IF REGION IS BL OR NS

      ICBL=0
```

```
      IF((I.GT.IV1.AND.I.LT.IB1).OR.(I.GT.IB2.AND.I.LT.IV2)) ICBL=1
C..CONSTRUCT MATRIX COEFFICIENTS FOR INTERIOR VORTICITY SOLUTION
      DO 100 J=2,JL
      T1=H(I,J)*R2(J)/DT
      GA(I,J)=A4(J)
      GB(I,J)=T1-A4(J)-C4(J)
      GC(I,J)=C4(J)
      GD(I,J)=T1*VOROLD(I,J)
C..TURBULENT FLOW TERMS AND BOUNDARY LAYER TERMS
      IF(ICTUR.EQ.1 .AND. IOTUR(I).EQ.1) THEN
        JPP1=J+1
        IF(JPP1.GT.JR) JPP1=JR
        GA(I,J)=GA(I,J)+A4(J)*EDDY(I,JPP1)/VSC
        GB(I,J)=GB(I,J)-(A4(J)+C4(J))*EDDY(I,J)/VSC
        GC(I,J)=GC(I,J)+C4(J)*EDDY(I,J-1)/VSC
      ENDIF
      IF(ICBL.EQ.1) THEN
        DIP1(I,J)=0.
        DIM1(I,J)=0.
      ELSE
        DIP1(I,J)=D5(J)
        DIM1(I,J)=D5(J)
        GB(I,J)=GB(I,J)+2.*D5(J)
        IF(ICTUR.EQ.1 .AND. IOTUR(I).EQ.1) THEN
          DIP1(I,J)=DIP1(I,J)+D5(J)*EDDY(IPP1,J)/VSC
          DIM1(I,J)=DIM1(I,J)+D5(J)*EDDY(IM1,J)/VSC
          GB(I,J)=GB(I,J)+2.*D5(J)*EDDY(I,J)/VSC
        ENDIF
      ENDIF
C..DISCRETIZATION OF CONVECTION TERM IN RADIAL DIRECTION
      VR=V(I,J+1)
      VL=V(I,J)
      A2VR=A2(J)*VR
      C2VL=C2(J)*VL
      IF(VR.LT.0.) THEN
        GA(I,J)=GA(I,J)+A2VR
      ELSE
        GB(I,J)=GB(I,J)+A2VR
      ENDIF
      IF(VL.LT.0.) THEN
        GB(I,J)=GB(I,J)+C2VL
      ELSE
        GC(I,J)=GC(I,J)+C2VL
      ENDIF
C..DISCRETIZATION OF CONVECTION TERM IN TANGENTIAL DIRECTION
      UL=0.25*(U(IM1,J+1)+U(IM1,J)+U(I,J+1)+U(I,J))
      UR=0.25*(U(I,J+1)+U(I,J)+U(IPP1,J+1)+U(IPP1,J))
      DTUL=UL/DTET
      DTUR=UR/DTET
      IF(UL.LT.0.) THEN
        GB(I,J)=GB(I,J)-DTUL
      ELSE
        DIM1(I,J)=DIM1(I,J)+DTUL
      ENDIF
      IF(UR.LT.0.) THEN
        DIP1(I,J)=DIP1(I,J)-DTUR
```

```
      ELSE
        GB(I,J)=GB(I,J)+DTUR
      ENDIF
  100 CONTINUE

C..NEUMANN TYPE B.C

      IF(JL.GE.JR) GB(I,JL)=GB(I,JL)+GA(I,JL)
  101 CONTINUE
      KKK=0
      ICMR=2
      DMAXP=100.
      URB=URBI

C..VORTICITY CONVERGENCE LOOP

  500 CONTINUE
      KKK=KKK+1

C..BOUNDARY CONDITION FOR KINETICS IN TURBULENCE

      IF(ICTUR.EQ.2) THEN
        DO 110 I=1,IM
        IF(IOTUR(I).EQ.1) THEN
            IN=I
            IF(IN.GT.KFC) IN=IM-I+2
            YP2=YN(IN,2)*USTAR(I)/VSC
            IF(YP2.GT.5.) THEN
              IOTUR(I)=0
            ELSE
              VOR(I,2)=VOR(I,1)
            ENDIF
        ENDIF
  110   CONTINUE
      ENDIF

C..SOLVE FOR EACH ZONE; USE ONLY HALF OF GRID IF SYMMETRIC FLOW

      IF(ALP.NE.0.) THEN
        CALL VORTY(IB1,IB2,1,ICS,0,KODE)
        IF((IV1+1).LT.IB1) CALL VORTY(IB1-1,IV1,-1,ICB1,1,KODE)
        IF(IV2.GT.IB2) CALL VORTY(IB2+1,IV2,1,ICB2,1,KODE)
        CALL VORTY(IV2,IV1R,1,ICV,0,KODE)
      ELSE
        CALL VORTY(IM2+2,IB2,1,ICS,0,KODE)
        CALL VORTY(IB2+1,IV2,1,ICB2,1,KODE)
        CALL VORTY(IV2,IM,1,ICV,0,KODE)
        DO 120 I=2,IM2
        II=IM+2-I
        DO 120 J=2,N
  120   VOR(I,J)=-VOR(II,J)
      ENDIF
      IF(KODE.GT.0) RETURN

C..DETERMINE VOR. EXTENT AT EACH RADIAL LINE IN N-S REGION
C  AND STORE IN ARRAY KEN

      DO 130 I=1,IM
      DO 129 J=N,2,-1
  129 IF(ABS(VOR(I,J)).GT.WMIN) GO TO 130
  130 KEN(I)=J

C..DETERMINE FOURIER COEFFICIENTS OF VORTICITY AND EVALUATE
C  SURFACE VORTICITY BY UNDER-RELAXATION TECHNIQUE
```

```
      DO 140 I=1,IM
      JL=KEN(I)
      JLP=JL+1
      DO 139 J=1,JL
  139 GAM(I,J)=H(I,J)*VOR(I,J)/FLOAT(IM2)
      IF(JLP.LE.N) THEN
        DO 138 J=JLP,N
  138   GAM(I,J)=0.
      ENDIF
  140 CONTINUE
      CALL FOCFT
      CALL WSURF
      MI=0
      MWI=0
      WMAX=0.
      DMAX=0.
      W1=0.
      DO 150 I=1,IM
        W1=AA(1,1)*.5
        DO 149 K=2,IM2
  149   W1=W1+AA(1,K)*CS(K,I)+BB(1,K)*SN(K,I)
        W1=W1+AA(1,KFC)*CS(KFC,I)*.5
        W1=W1/H(I,1)
        UR1=URB*H(I,1)**URBP
        WW=W1*UR1+(1.-UR1)*VOR(I,1)
        AWW=ABS(WW)
        IF(AWW.LT.10.) AWW=10.
        DW=ABS(WW-VOR(I,1))/AWW
        IF(DW.GE.DMAX) DMAX=DW
        IF(DW.GE.DMAX) MI=I
        IF(AWW.GE.WMAX) WMAX=AWW
        IF(AWW.GE.WMAX) MWI=I
        VOR(I,1)=WW
  150 CONTINUE

C..ADJUST UNDER-RELAXATION PARAMETER URB.
C  EXIT KNTCS IF CONVERGED, CONTINUE ITERATIONS IF NOT.
C  ABORT IF MAXIMUM ITERATIONS EXCEEDED.

      IF(KKK.EQ.ICMR) THEN
        DDM=(DMAXP-DMAX)/DMAX
        IF(DDM.LT.0.) URB=0.90*URB
        IF(DDM.LT.0.1.AND.DDM.GT.0.) URB=1.08*URB
        IF(DDM.GT.0.1) URB=1.05*URB
        IF(URB.GT.0.65) URB=0.65
        ICMR=ICMR+2
      ENDIF
      DMAXP=DMAX
      IF(MOD(NT,NTOUT).EQ.0 .OR. LOOP.EQ.NTMAX)
     + WRITE(6,12)KKK,DMAX,MI,VOR(MI,1),WMAX,MWI,URB,ICS,ICB1,ICB2,ICV
      IF(DMAX.LE.DRMX) RETURN
      IF(KKK.LT.KMAX) GO TO 500

C..NO CONVERGENCE OCCURED; RETURN TO MAIN WITH KODE=1

      KODE=1
      IF(MOD(NT,NTOUT).NE.0 .OR. LOOP.NE.NTMAX) THEN
        WRITE(6,10)
        WRITE(6,12)KKK,DMAX,MI,VOR(MI,1),WMAX,MWI,URB,ICS,ICB1,ICB2,ICV
      ENDIF
      RETURN
   10 FORMAT(//,25X,'SURFACE VORTICITY INFORMATION@D',20X,
     1  'INTERIOR VORTICITY INFORMATION@D'/' ITER',6X,
```

```
  2  'REL. MAX.      I     VOR(I,1)',9X,'MAX. VAL.     I     URB    ',
  3    9X,'ITSØD  ST,BL1,BL2,NS'/,13X,'DIFF.',29X,'OF VOR.')
 12 FORMAT(I4,4X,E13.6,I4,1X,E13.5,6X,E11.5,I4,E9.2,16X,4I3)
    END
```

```
      SUBROUTINE WSURFT
C
C  *********************************************************
C    EVALUATE BOUNDARY VORTICITIES IN TURBULENT FLOW REGION
C
C    CALLS      @D USTARF
C
C    CALLED BY @D KNTCS
C  *********************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (KFC1=41)

      COMMON/SCALE/H(IDIM,JDIM)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/TUR1/USTAR(IDIM),EDDY(IDIM,JDIM),IOTUR(IDIM),
     1            YN(KFC1,JDIM),IOT,IOTB
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/GRD/IM,IM2,KFC,JR,N

C..EVALUATE USTAR

      OMG2=2.*OMG
      USTAR(1)=0.
      DO 100 I=2,IM
      IN=I
      IF(IN.GT.KFC) IN=IM-I+2
      GM=(VOR(I,1)-OMG2)*YN(IN,1)
      AGM=ABS(GM)
      YP=SQRT(AGM*YN(IN,1)/VSC)
      USTAR(I)=AGM/YP
      IF(YP.GT.5.) USTAR(I)=USTARF(USTAR(I),AGM,YN(IN,1),5.,-3.05,YP)
      IF(YP.GT.30.) USTAR(I)=USTARF(USTAR(I),AGM,YN(IN,1),2.5,5.5,YP)
  100 CONTINUE
      RETURN
      END
      FUNCTION USTARF(USTR,UV,YN,A,C,YP)
C
C  *********************************************************
C    EVALUATE USTAR ITERATIVELY IN INERTIAL LAYER
C
C    CALLS      @D NONE
C
C    CALLED BY @D WSURFT
C  *********************************************************
C
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      DO 100 I=1,20
        YP=YN*USTR/VSC
        USTARF=UV/(A*ALOG(YP)+C)
        ABSER=ABS((USTARF-USTR)/USTARF)
        IF(ABSER.LT.0.005) RETURN
        USTR=USTARF
  100 CONTINUE
      WRITE(6,10) YP, USTR,ABSER
   10 FORMAT(2X,'NO CONVERGENCE IN USTARF. YP,USTAR,ABSER@D',3F9.5)
      STOP
      END
```

```
      SUBROUTINE EDDYS
C
C   *****************************************************************
C    EVALUATE EDDY VISCOSITY
C
C    CALLS      @D NONE
C
C    CALLED BY @D KNTCS
C   *****************************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41)
      PARAMETER (INOR1=20)

      COMMON/SCALE/H(IDIM,JDIM)
      COMMON/IO/LOOP,NT,NTMAX,NTOUT
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/TUR1/USTAR(IDIM),EDDY(IDIM,JDIM),IOTUR(IDIM),
     1            YN(KFC1,JDIM),IOT,IOTB
      COMMON/TUR2/INOR(INOR1,JDIM),IWK(JDIM,JDIM)
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/VEL/U(IDIM,JP1),V(IDIM,JP1)

      IMB=IM+2
      DO 105 II=2,IM
        I=II
        IOTUR(I)=0
        FMAX=0.
        YMAX=0.
        VDMX=0.
        JL=KIN(I)+1
        IF(JL.GT.JR) JL=JR
        IF(KIN(I+1).GT.JL) JL=KIN(I+1)
        IF(KIN(I-1).GT.JL) JL=KIN(I-1)

C..CALCULATE TERMS INDEPENDENT OF Y AND INNER VISC.
C  INCLUDE CORRECTION FOR THE NORMAL DIRECTION

        DO 100 J=2,JL
          IF(II.LT.INOR1+1) I=INOR(II,J)
          IF(II.GT.JP1) I=IMB-INOR(IMB-II,J)
          IN=I
          IF(I.GT.KFC) IN=IMB-I
          AVOR=ABS(VOR(I,J))
          YP=USTAR(II)*YN(IN,J)/VSC
          IF(YP.GT.500.) THEN
            T1=YN(IN,J)
          ELSE
            T1=YN(IN,J)*(1.-EXP(-YP/26.))
          ENDIF
          EDDY(I,J)=0.16*T1*T1*AVOR
          FY=AVOR*T1
          IF(FY.GT.FMAX) THEN
            FMAX=FY
            YMAX=YN(IN,J)
          ENDIF
          UV=0.5*(U(I,J+1)+U(I,J))
          VV=0.5*(V(I,J+1)+V(I,J))
          VSPHY=(UV*UV+VV*VV)/H(I,J)
          IF(VSPHY.GT.VDMX) VDMX=VSPHY
  100   CONTINUE
```

```
C..OUTER EDDY VISCOSITY

      IF(FMAX.EQ.0.) GO TO 105
      FWAKE=FMAX*YMAX
      CFWAKE=0.25*YMAX*VDMX/FMAX
      IF(CFWAKE.LT.FWAKE) FWAKE=CFWAKE
      ICED=0
      DO 103 J=2,JL
        IF(II.LT.INOR1+1) I=INOR(II,J)
        IF(II.GT.JP1) I=IMB-INOR(IMB-II,J)
        IN=I
        IF(I.GT.KFC) IN=IMB-I
        FKLEB=1./(1.+5.5*(0.3*YN(IN,J)/YMAX)**6.)
        CEDDY=0.0168*1.1*FWAKE*FKLEB
        IF(ICED.EQ.0 .AND. CEDDY.LT.EDDY(I,J)) THEN
          ICED=1
          IF(CEDDY/VSC.GT.0.) IOTUR(II)=1
        ENDIF
        IF(ICED.EQ.1) EDDY(I,J)=CEDDY
  103   CONTINUE
  105 CONTINUE

C..MAKE SURE FLOW REMAINS TURBULENT AFTER TRANSITION

      IOT=0
      IOTB=0
      DO 110 I=KFC,1,-1
        IB=IMB-I
        IF(IB.GT.IM) IB=IM
        IF(IOTUR(I).EQ.1 .AND. IOT.EQ.0) IOT=I
        IF(IOT.NE.0) IOTUR(I)=1
        IF(IOTUR(IB).EQ.1 .AND. IOTB.EQ.0) IOTB=IB
        IF(IOTB.NE.0) IOTUR(IB)=1
  110 CONTINUE

C..ASSIGN EDDY VISCOSITY IN THE WAKE ALONG WAKE GRID

      DO 120 JW=2,JR-1
        IW=INOR(2,JW)
        IBW=IMB-IW
        DO 119 J=JW,JR
          I=IWK(JW,J)
          IF(I.EQ.INOR(2,J)) GO TO 116
  115     IB=IMB-I
          IF(I.EQ.1) IB=1
          EDDY(I,J)=EDDY(IW,JW)
          EDDY(IB,J)=EDDY(IBW,JW)
  116     I=I-1
          IF(IWK(JW-1,J).LT.I) GO TO 115
  119   CONTINUE
  120 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VORTY(IS,IL,INC,IC,ICBL,KODE)
C
C   ********************************************************
C    CALCULATE VORTICITY BY USING LINE-
C    RELAXATION METHOD ON EACH RADIAL LINE
C
C    CALLS      @D TRID
C
C    CALLED BY @D KNTCS
C   ********************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (KFC1=41)

      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/TUR1/USTAR(IDIM),EDDY(IDIM,JDIM),IOTUR(IDIM),
     1            YN(KFC1,JDIM),IOT,IOTB
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/UNF/URR,URF,DFMX,NCC
      COMMON/SGA/GA(IDIM,JDIM),GB(IDIM,JDIM),GC(IDIM,JDIM)
      COMMON/RHS/GD(IDIM,JDIM),DIP1(IDIM,JDIM),DIM1(IDIM,JDIM)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/SOLV/SUB(JDIM),DIAG(JDIM),SUP(JDIM),RHS(JDIM)

      IF(KODE.EQ.1) RETURN
      DO 100 IC=1,NCC
      WMAX=0.
      DMAX=0.
      DO 110 II=IS,IL,INC
        I=II
        IPP1=I+1
        IM1=I-1
        IF(I.GT.IM) I=II-IM
        IF(IPP1.GT.IM) IPP1=IPP1-IM
        IF(IM1.GT.IM) IM1=IM1-IM
        JI=2
        IF(ICTUR.EQ.2 .AND. IOTUR(I).EQ.1) JI=3
        JIM=JI-1
        JL=KIN(I)
C..CONSTRUCT TRIDIAGONAL MATRIX AND SOLVE

        DO 120 J=JI,JL
          J1=J-JIM
          SUP(J1)=GA(I,J)
          DIAG(J1)=GB(I,J)
          SUB(J1)=GC(I,J)
          RHS(J1)=GD(I,J)+DIP1(I,J)*VOR(IPP1,J)+DIM1(I,J)*VOR(IM1,J)
  120   CONTINUE
        RHS(1)=RHS(1)-SUB(1)*VOR(I,JIM)
        CALL TRID(J1)
C..UNDER-RELAX THE RESULT OF THE MATRIX SOLUTION

        DO 130 J=JI,JL
          IF(ICBL .EQ. 1) THEN
            WW=RHS(J-JIM)
          ELSE
            WW=URF*VOR(I,J)+URR*RHS(J-JIM)
            WABS=ABS(WW)
            DD=ABS(WW-VOR(I,J))
            IF(DD.GE.DMAX) DMAX=DD
            IF(WABS.GT.WMAX) WMAX=WABS
```

```
            ENDIF
            VOR(I,J)=WW
  130     CONTINUE
  110 CONTINUE

C..IF BOUNDARY LAYER ZONE, RETURN TO KNTCS AFTER SOLVING EXPLICITLY.
C  ELSE ITERATE FOR CONVERGENCE AND RETURN WHEN CONVERGED OR WHEN
C  MAX NUMBER OF ITERATIONS IS EXCEEDED.

      IF(ICBL.EQ.1) RETURN
      IF(DMAX/WMAX.LE.DFMX) RETURN
  100 CONTINUE
      WRITE(6,10) II,DMAX
      KODE=1
      RETURN
   10 FORMAT(2X,'NO TRID CONVERGENCE AT I=',I4,'   DMAX=',E10.3)
      END
```

```
      SUBROUTINE TRID(N)
C
C   *******************************************************************
C     TRIDIAGONAL MATRIX SOLVER
C
C     CALLS      ØD NONE
C
C     CALLED BY ØD VORTY
C   *******************************************************************
C
      PARAMETER (JDIM=60)
      COMMON/SOLV/SUB(JDIM),DIAG(JDIM),SUP(JDIM),RHS(JDIM)

      DO 100 J=2,N
      RATIO=-SUB(J)/DIAG(J-1)
      DIAG(J)=DIAG(J)+RATIO*SUP(J-1)
      RHS(J)=RHS(J)+RATIO*RHS(J-1)
  100 CONTINUE
      RHS(N)=RHS(N)/DIAG(N)
      DO 110 J=N-1,1,-1
  110 RHS(J)=(RHS(J)-SUP(J)*RHS(J+1))/DIAG(J)
      RETURN
      END
```

```
      SUBROUTINE FOCFT
C
C   ****************************************************
C    DETERMINE FOURIER COEFFICIENTS AT EACH RING
C
C    CALLS      ●D NONE
C
C    CALLED BY ●D VORTY
C                 CPVAL
C   ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (KFC1=41)

      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/SMT/SGMA(KFC1)
      COMMON/SCALE/H(IDIM,JDIM)
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/TRIG/CS(KFC1,IDIM),SN(KFC1,IDIM)
      COMMON/FM/GAM(IDIM,JDIM)

      DO 100 J=1,N
      DO 100 K=1,KFC
        BB(J,K)=0.0
        AA(J,K)=0.0
  100 CONTINUE
      DO 110 I=1,IM
        NI=KEN(I)
        DO 109 J=1,NI
          AA(J,1)=AA(J,1)+GAM(I,J)
          DO 109 K=2,KFC
          AA(J,K)=AA(J,K)+GAM(I,J)*CS(K,I)
          BB(J,K)=BB(J,K)+GAM(I,J)*SN(K,I)
  109   CONTINUE
  110 CONTINUE
      DO 120 J=2,N
      DO 120 K=2,KFC
        AA(J,K)=AA(J,K)*SGMA(K)
        BB(J,K)=BB(J,K)*SGMA(K)
  120 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WSURF
C
C   ****************************************************
C    COMPUTE SURFACE VORTICITY
C
C    CALLS     @D NONE
C
C    CALLED BY @D KNTCS
C   ****************************************************
C
      PARAMETER (IDIM=80, JDIM=60)
      PARAMETER (JP1=61)
      PARAMETER (KFC1=41,KFC2=42)

      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/SMT/SGMA(KFC1)
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/ABCDS/AS1(KFC1),BS1(KFC1),CS1(KFC1),DS1(KFC1)
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/COR/RP(JP1,KFC2),RL(JP1)
      COMMON/VLB/VORLB,NLB
      COMMON Q1(IDIM,JDIM),Q2(IDIM,JDIM),Q3(IDIM,JDIM),Q4(IDIM,JDIM)

C..DETERMINE FOURIER COEFFS. OF BOUNDARY VORTICITY, AA(1,M),BB(1,M)
C  FOR M.GE.4

      PI=4.*ATAN(1.)
      DO 100 K=4,KFC
        CW1=0.
        DW1=0.
        DO 101 J=2,N
          W1=1./RP(J,K-3)-1./RP(J+1,K-3)
          CW1=CW1+AA(J,K)*W1
          DW1=DW1+BB(J,K)*W1
  101   CONTINUE
        W2=1./(1./RP(1,K-3)-1./RP(2,K-3))
        BB(1,K)=W2*((AS1(K)-DS1(K))*FLOAT(K-3)-DW1)
        AA(1,K)=-W2*((CS1(K)+BS1(K))*FLOAT(K-3)+CW1)
  100 CONTINUE

C..DETERMINE AA(1,K),BB(1,K), FOR K.LE.3

      CW=0.
      CW1=0.
      CW2=0.
      DW1=0.
      DW2=0.
      DO 130 J=2,N
        CW1=CW1+AA(J,2)*(RP(J+1,1)-RP(J,1))
        DW1=DW1+BB(J,2)*(RP(J+1,1)-RP(J,1))
        CW2=CW2+AA(J,3)*(RL(J+1)-RL(J))
        DW2=DW2+BB(J,3)*(RL(J+1)-RL(J))
  130 CONTINUE
      DO 140 J=2,NLB
  140 CW=CW+AA(J,1)*(RP(J+1,2)-RP(J,2))
      AA(1,1)=-(4.*AF/PI*OMG+CW+2.*VORLB/PI)/(RP(2,2)-RP(1,2))
      AA(1,2)=(2.*VI-CS1(2)-BS1(2)-CW1)/(RP(2,1)-RP(1,1))
      BB(1,2)=(AS1(2)-DS1(2)-2.*UI-DW1)/(RP(2,1)-RP(1,1))
      AA(1,3)=-(CS1(3)+BS1(3)+CW2)/RL(2)
      BB(1,3)=(AS1(3)-DS1(3)-DW2)/RL(2)
      DO 150 K=2,KFC
        AA(1,K)=AA(1,K)*SGMA(K)
        BB(1,K)=BB(1,K)*SGMA(K)
  150 CONTINUE
```

```
RETURN
END
```

```
      SUBROUTINE KMTCS
C
C  ******************************************************
C    KINEMATICS OF THE PROBLEM
C
C    CALLS     @D NONE
C
C    CALLED BY @D ZONST
C  ******************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41,KFC2=42)

      DIMENSION VTOTPH(IDIM,JDIM),VTOTCO(IDIM,JDIM)

      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/SMT/SGMA(KFC1)
      COMMON/ABCDS/AS1(KFC1),BS1(KFC1),CS1(KFC1),DS1(KFC1)
      COMMON/RHS/A(JP1,KFC1),B(JP1,KFC1),C(JP1,KFC1),D(JP1,KFC1)
      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/COR/RP(JP1,KFC2),RL(JP1)
      COMMON/VEH/UC(IP1,JP1),VC(IP1,JP1)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/VEL/U(IDIM,JP1),V(IDIM,JP1),HSTAR(IDIM,JDIM)
      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/TRIG/CS(KFC1,IDIM),SN(KFC1,IDIM)
      COMMON Q1(IDIM,JDIM),Q2(IDIM,JDIM),Q3(IDIM,JDIM),Q4(IDIM,JDIM)
      COMPLEX HSTAR,VTOTCO,VTOTPH

      NP=N+1
      DO 100 K=1,KFC
        A(1,K)=AS1(K)
        B(1,K)=BS1(K)
        C(1,K)=CS1(K)
        D(1,K)=DS1(K)
  100 CONTINUE

      DO 500 J=2,NP
      JM1=J-1

C..BOUNDARY VELOCITY CONTRIBUTIONS

      W1=.5/RP(J,2)
      C(J,2)=W1*(C(1,2)-B(1,2))+VI
      D(J,2)=W1*(D(1,2)+A(1,2))-UI
      A(J,1)=A(1,1)/RP(J,1)
      A(J,2)=W1*(D(1,2)+A(1,2))+UI
      B(J,2)=W1*(B(1,2)-C(1,2))+VI
      DO 110 K=3,KFC
        W1=.5/RP(J,K)
        C(J,K)=W1*(C(1,K)-B(1,K))
        D(J,K)=W1*(D(1,K)+A(1,K))
        A(J,K)=D(J,K)
        B(J,K)=-C(J,K)
  110 CONTINUE

C..INTEGRATE ALONG RADIAL AXIS FOR AA(J,K), BB(J,K); (K.GE.4)

      DO 120 K=4,KFC
        CW=0.
        CW1=0.
        DW=0.
```

```
      DW1=0.
      DO 119 JJ=1,JM1
        W1=RP(JJ+1,K+1)-RP(JJ,K+1)
        CW=AA(JJ,K)*W1+CW
        DW=BB(JJ,K)*W1+DW
119   CONTINUE
      IF(J.LT.NP) THEN
        DO 118 JJ=J,N
          W1=1./RP(JJ,K-3)-1./RP(JJ+1,K-3)
          CW1=AA(JJ,K)*W1+CW1
          DW1=BB(JJ,K)*W1+DW1
118     CONTINUE
      ENDIF
      W2=.5/(FLOAT(K+1)*RP(J,K))
      CW2=W2*CW
      DW2=W2*DW
      W3=.5*RP(J,K-2)/FLOAT(K-3)
      CW3=W3*CW1
      DW3=W3*DW1
      C(J,K)=C(J,K)+CW2-CW3
      D(J,K)=D(J,K)+DW2-DW3
      A(J,K)=A(J,K)+DW2+DW3
      B(J,K)=B(J,K)-CW2-CW3
120 CONTINUE

C..COMPUTE AA(J,K), BB(J,K); (K.LE.3)

      CW=0.
      CW1=0.
      CW2=0.
      CW3=0.
      CW4=0.
      DW1=0.
      DW2=0.
      DW3=0.
      DW4=0.
      DO 130 JJ=1,JM1
        CW=CW+AA(JJ,1)*(RP(JJ+1,2)-RP(JJ,2))
        CW1=CW1+AA(JJ,2)*(RP(JJ+1,3)-RP(JJ,3))
        DW1=DW1+BB(JJ,2)*(RP(JJ+1,3)-RP(JJ,3))
        CW2=CW2+AA(JJ,3)*(RP(JJ+1,4)-RP(JJ,4))
        DW2=DW2+BB(JJ,3)*(RP(JJ+1,4)-RP(JJ,4))
130 CONTINUE
      IF(J.LT.NP) THEN
        DO 140 JJ=J,N
          CW3=CW3+AA(JJ,2)*(RP(JJ+1,1)-RP(JJ,1))
          DW3=DW3+BB(JJ,2)*(RP(JJ+1,1)-RP(JJ,1))
          CW4=CW4+AA(JJ,3)*(RL(JJ+1)-RL(JJ))
          DW4=DW4+BB(JJ,3)*(RL(JJ+1)-RL(JJ))
140     CONTINUE
      ENDIF
      C(J,1)=(C(1,1)+.5*CW)/RP(J,1)
      CW3=.5*CW3
      DW3=.5*DW3
      W2=1./(6.*RP(J,2))
      CW1=CW1*W2
      DW1=DW1*W2
      C(J,2)=C(J,2)+CW1-CW3
      D(J,2)=D(J,2)+DW1-DW3
      A(J,2)=A(J,2)+DW1+DW3
      B(J,2)=B(J,2)-CW1-CW3
      W2=1./(8.*RP(J,3))
      CW2=CW2*W2
      DW2=DW2*W2
```

```
      W2=.5*RP(J,1)
      CW4=CW4*W2
      DW4=DW4*W2
      C(J,3)=C(J,3)+CW2-CW4
      D(J,3)=D(J,3)+DW2-DW4
      A(J,3)=A(J,3)+DW2+DW4
      B(J,3)=B(J,3)-CW2-CW4
  500 CONTINUE

      DO 150 K=2,KFC
      DO 150 J=2,N
        A(J,K)=A(J,K)*SGMA(K)
        B(J,K)=B(J,K)*SGMA(K)
        C(J,K)=C(J,K)*SGMA(K)
        D(J,K)=D(J,K)*SGMA(K)
  150 CONTINUE

C..ABOVE COEFFICIENTS ARE FOR V(RHO) & V(PHI) IN NON-ROTATING
C  FRAME OF REFERENCE IN COMPUTATIONAL PLANE.
C  CALCULATE VELOCITIES WITH THESE VALUES.

      DO 160 I=1,IM
        JL=KIN(I)+1
        IF(ICPL.EQ.0 .AND. NPL.GT.JL) JL=NPL
        IPP1=I+1
        IM1=I-1
        IF(I.EQ.1) IM1=IM
        IF(I.EQ.IM) IPP1=1
        IF(KIN(IPP1).GE.JL) JL=KIN(IPP1)+1
        IF(KIN(IM1).GE.JL) JL=KIN(IM1)+1
        DO 159 J=2,JL
        U(I,J)=C(J,1)*.5
        V(I,J)=A(J,1)*.5
        DO 158 K=2,IM2
        U(I,J)=U(I,J)+C(J,K)*CS(K,I)+D(J,K)*SN(K,I)
        V(I,J)=V(I,J)+A(J,K)*CS(K,I)+B(J,K)*SN(K,I)

  158   CONTINUE

        U(I,J)=U(I,J)+C(J,KFC)*CS(KFC,I)*.5
        V(I,J)=V(I,J)+A(J,KFC)*CS(KFC,I)*.5

  159   CONTINUE


C..TRANSFORM VELOCITY COMPONENTS TO ROTATING FRAME OF REFERENCE
C  FOR USE IN KNTCS

        DO 157 J=2,JL
          W1=V(I,J)*CS(2,I)-U(I,J)*SN(2,I)
          W2=V(I,J)*SN(2,I)+U(I,J)*CS(2,I)
          W1=W1+OMG*UC(I,J)
          W2=W2+OMG*VC(I,J)
          IF( J.LE.60 )THEN
            VTOTCO(I,J) = CMPLX (W1,W2)
            VTOTPH(I,J) = VTOTCO(I,J)/HSTAR(I,J)
            Q2(I,J) = REAL(VTOTPH(I,J))
            Q3(I,J) = AIMAG(VTOTPH(I,J))
          ENDIF
          U(I,J)=W2*CS(2,I)-W1*SN(2,I)
          V(I,J)=W1*CS(2,I)+W2*SN(2,I)

  157   CONTINUE
  160 CONTINUE
```

```
RETURN
END
```

```
      SUBROUTINE CPVAL
C
C   ****************************************************
C    CALCULATE CP VALUES BY CP INTEGRAL
C
C    CALLS      @D COED
C                  FOCFT
C
C    CALLED BY @D ZONST
C   ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (KFC1=41)
      PARAMETER (ICP=40)

      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/TRIG/CS(KFC1,IDIM),SN(KFC1,IDIM)
      COMMON/VV/VOR(IDIM,JDIM),VOROLD(IDIM,JDIM)
      COMMON/COE/AF,UI,VI,OMG,VSC,NPL,ICTUR,ICST,ICPL
      COMMON/SMT/SGMA(KFC1)
      COMMON/VEL/U(IDIM,JP1),V(IDIM,JP1)
      COMMON/VTEXT/KIN(IDIM),KEN(IDIM)
      COMMON/CPC/CCP(KFC1,JDIM),CP(IP1)
      COMMON/WFC/AA(JDIM,KFC1),BB(JDIM,KFC1)
      COMMON/RHS/A(JP1,KFC1),B(JP1,KFC1),C(JP1,KFC1),D(JP1,KFC1)
      COMMON/FM/GAM(IDIM,JDIM)
      COMMON/CPW/AW(IDIM),BW(IDIM),CW(ICP),DW(ICP),EW(ICP),FW(ICP),
     1           AA2(ICP),BB2(ICP),CC2(ICP),DD2(ICP)
      COMMON Q1(IDIM,JDIM),Q2(IDIM,JDIM),Q3(IDIM,JDIM),Q4(IDIM,JDIM)
      DIMENSION UCC(JDIM,KFC1),UCS(JDIM,KFC1),VCC(JDIM,KFC1),
     1          VCS(JDIM,KFC1),FC(KFC1),FS(KFC1)

      DO 105 I=1,IM
        JL=KEN(I)
        JLP=JL+1
        DO 100 J=1,JL
  100   GAM(I,J)=VOR(I,J)/FLOAT(IM2)
        IF(JLP.LE.N) THEN
          DO 103 J=JLP,N
  103     GAM(I,J)=0.
        ENDIF
  105 CONTINUE
      DO 110 I=1,KFC
        FC(I)=0.
        FS(I)=0.
  110 CONTINUE
      CALL FOCFT
      DO 120 I=1,N
      I1= I+1
        DO 115 J=2,KFC
        J1= J-1
        AW(J1)=AA(I,J)
        BW(J1)=BB(I,J)
        CW(J1)=.5*( C(I,J) + C(I1,J) )
        DW(J1)=.5*( D(I,J) + D(I1,J) )
        EW(J1)=.5*( A(I,J) + A(I1,J) )
        FW(J1)=.5*( B(I,J) + B(I1,J) )
  115 CONTINUE
      AO=AA(I,1)*.5
      CO=.25*( C(I,1) + C(I1,1) )
      AOO=.25*( A(I,1) + A(I1,1) )
      CALL COED(AO,CO,AOO,UO)
      UCC(I,1)=2.*UO
```

```
      DO 116 J=2,KFC
        J1=J-1
        UCC(I,J)=AA2(J1)
        UCS(I,J)=BB2(J1)
        VCC(I,J)=CC2(J1)
        VCS(I,J)=DD2(J1)
116   CONTINUE
120   CONTINUE
      DO 130 J=1,N
        DO 129 I=2,IM2
          FC(I)=FC(I) + (VCS(J,I)-UCC(J,I)) *CCP(I,J)
          FS(I)=FS(I) + (UCS(J,I)+VCC(J,I)) *CCP(I,J)
129     CONTINUE
        FC(KFC)=FC(KFC) - UCC(J,KFC)*CCP(KFC,J)
        FC(1)=FC(1) + UCC(J,1)*CCP(1,J)
130   CONTINUE
        DO 131 J=1,JDIM
          DO 131 I=1,IDIM
          Q4(I,J)=CCP(I,J)/.4+.5*(U(I,J)**2+V(I,J)**2)
131   CONTINUE
        DO 140 I=2,IM2
        FC(I)=FC(I) + VSC*BB(1,I)
        FS(I)=FS(I) + VSC*AA(1,I)
140   CONTINUE
        DO 150 J=2,KFC
        FC(J)=FC(J)*SGMA(J)
        FS(J)=FS(J)*SGMA(J)
150   CONTINUE
      DO 160 I=1,IM
        CP(I)=1.-FC(1)
        DO 159 L=2,IM2
159     CP(I)=CP(I) + 2.*( FC(L)*CS(L,I)-FS(L)*SN(L,I) )
        CP(I)=CP(I) + FC(KFC)*CS(KFC,I)
160   CONTINUE
      CP(IM+1)=CP(1)
      RETURN
      END
```

```
      SUBROUTINE COED(AO,CO,AOO,UO)
C
C  ****************************************************************
C    MULTIPLICATION OF TWO FOURIER SERIES
C
C    CALLS     @D NONE
C
C    CALLED BY @D CPVAL
C  ****************************************************************
C
      PARAMETER (IDIM=80)
      PARAMETER (ICP=40)

      COMMON/GRD/IM,IM2,KFC,JR,N
      COMMON/CPW/AW(IDIM),BW(IDIM),CW(ICP),DW(ICP),EW(ICP),FW(ICP),
     1           AA2(ICP),BB2(ICP),CC2(ICP),DD2(ICP)

      NM1= IM2-1
      UO= CO*AO
      DO 100 I=1,IM2
  100 UO= UO + .5*(AW(I)*CW(I)+BW(I)*DW(I))
      DO 110 I=1,IM2
        II= I+I
        AA2(I)= AW(II)*CW(I) + BW(II)*DW(I)
        BB2(I)= BW(II)*CW(I) - AW(II)*DW(I)
        CC2(I)= AW(II)*EW(I) + BW(II)*FW(I)
        DD2(I)= BW(II)*EW(I) - AW(II)*FW(I)
  110 CONTINUE
      DO 120 J=1,NM1
        K4= IM2-J
        DO 119 K=1,J
          K1= IM2+1-K
          K2= K1-K4
          K3= K1+K4
          W1= AW(K2)+AW(K3)
          W2= BW(K2)+BW(K3)
          W3= AW(K2)-AW(K3)
          W4= BW(K3)-BW(K2)
          AA2(K4)= AA2(K4) + W1*CW(K1) + W2*DW(K1)
          BB2(K4)= BB2(K4) + W3*DW(K1) + W4*CW(K1)
          CC2(K4)= CC2(K4) + W1*EW(K1) + W2*FW(K1)
          DD2(K4)= DD2(K4) + W3*FW(K1) + W4*EW(K1)
  119   CONTINUE
  120 CONTINUE
      DO 130 J=1,NM1
        K4= J+1
        DO 129 K=1,J
          K2= K4-K
          K3= K4+K
          W1= AW(K2)+AW(K3)
          W2= BW(K3)-BW(K2)
          W3= AW(K2)-AW(K3)
          W4= BW(K2)+BW(K3)
          AA2(K4)= AA2(K4) + W1*CW(K) + W2*DW(K)
          BB2(K4)= BB2(K4) + W3*DW(K) + W4*CW(K)
          CC2(K4)= CC2(K4) + W1*EW(K) + W2*FW(K)
          DD2(K4)= DD2(K4) + W3*FW(K) + W4*EW(K)
  129   CONTINUE
  130 CONTINUE
      DO 140 I=1,IM2
        AA2(I)= .5*AA2(I) + AO*CW(I) + CO*AW(I)
        BB2(I)= .5*BB2(I) + AO*DW(I) + CO*BW(I)
        CC2(I)= .5*CC2(I) + AO*EW(I) + AOO*AW(I)
        DD2(I)= .5*DD2(I) + AO*FW(I) + AOO*BW(I)
```

```
140 CONTINUE
    AA2(IM2)= 2.*AA2(IM2)
    CC2(IM2)= 2.*CC2(IM2)
    RETURN
    END
```

```
4 .15  19.065 1
.0001 .0005 .0005 75 100
.4 .3 .6
10 35 50 74 45 40
.10 .10  50
 50  50  50
```

```
ICST,RF,ALPS,ICTUR
WMIN,DFMX,DRMX,KMAX,NCC
URBI,URBP,URR
IV1,IB1,IB2,IV2,NPL,NLB
DTI,DTINC,NTMAX
NTPL,NTOUT,NTLO
```

```
      PROGRAM LOADS

C  ********************************************************
C    LOADS COMPUTATIONS - DISSPLA VERSION
C    LAST REVISION 4-7-87
C
C    PRINCIPAL INVESTIGATOR : DR. J.C. WU
C    AUTHORS : MIKE PATTERSON, ISHMAEL TUNCER
C              GEORGIA INSTITUTE OF TECHNOLOGY
C              (404) 894-3028
C
C    TAPE3   : INPUT FROM GEOM
C    TAPE4   : INPUT FROM ZONST
C    TAPE5   : GENERAL INPUT
C    TAPE6   : GENERAL OUTPUT
C    TAPE10  : OUTPUT TO PLOT3
C
C    CALLS   : INTER
C  ********************************************************

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)

      DIMENSION WB(IP1),CP(IP1),G(IP1)
      DIMENSION XB(IDIM),YB(IDIM),XT(IDIM),YT(IDIM),FUN(IP1)
      COMMON/L1/DTET,IM,IM1

C       REWIND 3
C       REWIND 4
C       REWIND 5
C       REWIND 6
C       REWIND 10

      READ(5,*) NTOUT,NTS,NTL
      READ(3) AL,RE,DTET,XB,YB,XT,YT

      IM=IDIM
      JR=JDIM
      IM1=IP1
      JR1=JP1
      PI=4.*ATAN(1.)

C..READ SURFACE VORTICITY & PRESSURE COEFFICIENT

  100 READ(4,END=99) NT,NTPL,T,RF,WB,CP,OMG,OMGD,ALP
      IF(NT.LT.NTS) GO TO 100
      IF(NT.GT.NTL) GO TO 99
      ALPD=ALP*180./PI
      WRITE(6,60) NT,T,ALPD
      COSA=COS(ALP)
      SINA=SIN(ALP)

C..COMPUTE TOTAL CP AND OUTPUT VALUES

      DO 110 I=1,IM
  110 FUN(I)=OMGD*(XB(I)*YT(I)-YB(I)*XT(I))
      FUN(IM1)=FUN(1)
      CALL INTER(FUN,G,1,IM1)
      DO 120 I=2,IM1
      WW=OMG*OMG*(XB(I)*XB(I)+YB(I)*YB(I)-XB(1)*XB(1))
  120 CP(I)=CP(I)-G(I)+WW
      IF(MOD(NT,NTOUT).EQ.0) WRITE(6,61) (CP(I),I=1,IM)

C..COMPUTE TANGENTIAL FORCE COEFFICIENT FROM VISCOUS EFFECT
```

```
      DO 130 I=1,IM1
  130 WB(I)=WB(I)-2.*OMG
      DO 140 I=1,IM
  140 FUN(I)=WB(I)*XT(I)
      FUN(IM1)=FUN(1)
      CALL INTER(FUN,G,1,IM1)
      CTV=2.0*G(IM1)/RE

C..COMPUTE NORMAL FORCE COEFFICIENT FROM VISCOUS EFFECT

      DO 150 I=1,IM
  150 FUN(I)=WB(I)*YT(I)
      FUN(IM1)=FUN(1)
      CALL INTER(FUN,G,1,IM1)
      CNV=2.0*G(IM1)/RE

C..COMPUTE MOMENT COEFFICIENT FROM VISCOUS EFFECT

      DO 160 I=1,IM
  160 FUN(I)=WB(I)*(XB(I)*YT(I)-YB(I)*XT(I))
      FUN(IM1)=FUN(1)
      CALL INTER(FUN,G,1,IM1)
      CMV=2.0*G(IM1)/(RE*AL)
      WRITE(6,66) CNV,CTV,CMV

C..COMPUTE TANGENTIAL FORCE COEFFICIENT FROM PRESSURE DISTRIBUTION

      DO 170 I=1,IM
  170 FUN(I)=CP(I)*YT(I)
      FUN(IM1)=CP(IM1)*YT(1)
      CALL INTER(FUN,G,1,IM1)
      CTP=-G(IM1)/AL

C..COMPUTE NORMAL FORCE COEFFICIENT FROM PRESSURE DISTRIBUTION

      DO 180 I=1,IM
  180 FUN(I)=CP(I)*XT(I)
      FUN(IM1)=CP(IM1)*XT(1)
      CALL INTER(FUN,G,1,IM1)
      CNP=G(IM1)/AL

C..COMPUTE MOMENT COEFFICIENT FROM PRESSURE DISTRIBUTION

      DO 190 I=1,IM
  190 FUN(I)=CP(I)*(XB(I)*XT(I)+YB(I)*YT(I))
      FUN(IM1)=CP(IM1)*(XB(1)*XT(1)+YB(1)*YT(1))
      CALL INTER(FUN,G,1,IM1)
      CMP=G(IM1)/(AL*AL)

C..WRITE PRESSURE COEFFICIENTS AND COMPUTE TOTAL NORMAL, TANGENTIAL
C  AND MOMENT COEFFICIENTS

      WRITE(6,64) CNP,CTP,CMP
      CN=CNV+CNP
      CT=CTV+CTP
      CM=CMV+CMP

C..COMPUTE LIFT AND DRAG COEFFICIENTS

      CLP=CNP*COSA-CTP*SINA
      CDP=CNP*SINA+CTP*COSA
      CL=CN*COSA-CT*SINA
      CD=CN*SINA+CT*COSA
```

```
      WRITE(6,67) CLP,CDP,CMP
      WRITE(6,69) CL,CD,CM

C..WRITE DATA FOR PLOT3 PROGRAM AND RETURN TO BEGINNING OF PROGRAM

      WRITE(10) NT,NTPL,T,ALPD,RF,RE,CLP,CDP,CMP,CP
      GO TO 100

C..LAST OF DATA HAS BEEN READ

   99 WRITE(6,63)
   60 FORMAT(1H ,40(1H-),/' NT =',I4,',   T =',F7.4
     + ,'     AA =',F8.4/1X,40(1H-))
   61 FORMAT(//,' CP VALUES, I=1,IM :'//(10E13.6))
   63 FORMAT(//' ---: END OF FILE :---'//)
   64 FORMAT(' VALUES OF CNP,CTP,CMP -------',5X,3E16.8)
   66 FORMAT(//' VALUES OF CNV,CTV,CMV -------',5X,3E16.8)
   67 FORMAT(' PRESS  LOADS  CLP,CDP,CMP ---',5X,3E16.8)
   69 FORMAT(' TOTAL LOADS  CL,CD,CM -------',5X,3E16.8)
      STOP
      END
      SUBROUTINE INTER(A,AI,IS,IL)
C
C  ******************************************************
C    NUMERICAL QUADRATURE BY HIGH ORDER FORMULAS
C
C    CALLS     : NONE
C
C    CALLED BY : LOADS
C  ******************************************************
C
      DIMENSION A(IM1),AI(IM1)
      COMMON/L1/DTET,IM,IM1

      INC=1
      CI=DTET/24.
      IF(IL.LT.IS) THEN
        INC=-INC
        CI=-CI
      ENDIF
      AI(IS)=0.
      DO 100 I=IS+INC,IL,INC
        IM2=I-2*INC
        IP1=I+INC
        IF(IM2.LT.1.OR.IM2.GT.IM1) IM2=ABS(IM1-1-IM2)
        IF(IP1.LT.1.OR.IP1.GT.IM1) IP1=ABS(IM1-1-IP1)
        AI(I)=AI(I-INC)+CI*(-A(IM2)+13.*(A(I-INC)+A(I))-A(IP1))
  100 CONTINUE
      RETURN
      END
```

```
      PROGRAM PLOT1

C ******************************************************
C    GEOMETRY PLOTTING PROGRAM
C    LAST REVISION 4-1-87
C
C    PRINCIPAL INVESTIGATOR : DR. J.C. WU
C    AUTHORS : MIKE PATTERSON, ISHMAEL TUNCER
C              GEORGIA INSTITUTE OF TECHNOLOGY
C              (404) 894-3028
C
C    TAPE1   : INPUT FROM GEOMETRY PROGRAM
C    TAPE5   : GENERAL INPUT
C    TAPE11  : OUTPUT FOR PLOTTER
C
C    CALLS   : AXES, GRID, WAKE, ZONES
C ********************************************************

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (INOR1=20)

      COMMON/GRD/X(IDIM,JP1),Y(IDIM,JP1)
      COMMON/ZN/AL,JVL,JBL,JFL,IV1,IB1,IB2,IV2,ILINE(6),SCALE
      COMMON/WK/INOR(INOR1,JDIM),IWK(JDIM,JDIM)
      COMMON/PARAM/IM,JR
      DATA IRES/2/

C      REWIND(1)
C      REWIND(5)

      READ(1) AL,X,Y,INOR,IWK
      READ(5,*) IV1,IB1,IB2,IV2
      READ(5,*) IPLOT1,IPLOT2,IPLOT3,IPLOT4
      READ(5,*) JVL,JBL,JFL,SCALE,IDEV

      IM=IDIM
      JR=JDIM

      ILINE(1)=1
      ILINE(2)=IV1
      ILINE(3)=IB1
      ILINE(4)=IB2
      ILINE(5)=IV2
      ILINE(6)=IM

C. SET DEVICE

      CALL DIP(15,'PLOT1.DIP',9)

C      IF(IDEV.EQ.0) THEN
C        CALL EPIC(300.,0,8.25,10.75,11)
C      ELSE IF(IDEV.EQ.1) THEN
C       CALL PCLCMP
C       CALL CALCMP(IBUF,512,11)
C      ELSE IF(IDEV.EQ.2) THEN
C       CALL PVRSTC
C       CALL VRSTEC(IBUF,512,11)
C      ELSE IF(IDEV.EQ.3) THEN
C       CALL P4115
C       CALL TK4115(IRES)
C      ENDIF

C..CONSTRUCT EACH PLOT SPECIFIED
```

```
      IF(IPLOT1.EQ.1) THEN
        CALL AXES(JVL,0.,'VORTICITY GRID$')
        CALL GRID(IM,IM,JVL)
        CALL ENDPL(0)
      END IF
      IF(IPLOT2.EQ.1) THEN
        CALL AXES(JBL,AL/4.,'AIRFOIL AND BL GRID$')
        CALL THKFRM(.01)
        CALL FRAME
        CALL GRID(IM,IM,JBL)
        CALL ENDPL(0)
      END IF
      IF(IPLOT3.EQ.1) CALL ZONES
      IF(IPLOT4.EQ.1) CALL WAKE

      CALL DONEPL
      STOP
      END
      SUBROUTINE ZONES
C
C   ****************************************************
C     AIRFOIL AND FLOW ZONES
C
C     CALLS      : AXES, GRID
C
C     CALLED BY : PLOT1
C   ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)

      COMMON/GRD/X(IDIM,JP1),Y(IDIM,JP1)
      COMMON/ZN/AL,JVL,JBL,JFL,IV1,IB1,IB2,IV2,ILINE(6),SCALE
      COMMON/PARAM/IM,JR
      DIMENSION XX(IP1),YY(IP1)

      CALL AXES(JFL,AL/4.,'FLOW ZONES$')
      CALL THKFRM(.01)
      CALL FRAME
      CALL GRID(0,IM,5)
      CALL HEIGHT(.1)

C..PLOT AND NUMBER THE DEMARCATION LINES

      DO 110 I=1,6
      DO 109 J=1,JFL
        XX(J)=X(ILINE(I),J)
        YY(J)=Y(ILINE(I),J)
  109 CONTINUE
      CALL CURVE(XX,YY,JFL,0)
      CALL RLINT(ILINE(I),XX(JFL),Y(ILINE(I),JFL+1))
  110 CONTINUE
      CALL HEIGHT(.14)
      CALL ENDPL(0)
      RETURN
      END
      SUBROUTINE WAKE
C
C   ****************************************************
C     AIRFOIL AND WAKE GRID
C
C     CALLS      : AXES, GRID
C
```

```
C    CALLED BY : PLOT1
C    ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (INOR1=20)

      COMMON/GRD/X(IDIM,JP1),Y(IDIM,JP1)
      COMMON/ZN/AL,JVL,JBL,JFL,IV1,IB1,IB2,IV2,ILINE(6),SCALE
      COMMON/WK/INOR(INOR1,JDIM),IWK(JDIM,JDIM)
      COMMON/PARAM/IM,JR
      DIMENSION XX(IP1),YY(IP1)

C..PLOT AXES AND AIRFOIL

      CALL AXES(JVL,AL,'WAKE GRID$')
      CALL GRID(0,IM,5)

C..PLOT VORTICITY GRID UNDERNEATH WAKE GRID

      CALL DOT
      CALL THKCRV(.001)
      CALL GRID(IM/4+1,IM/4+1,JR)
      CALL RESET('DOT')
      CALL RESET('THKCRV')

C..PLOT STREAMWISE LINES

      DO 110 JN=1,JR-1
      JJ=0
      DO 109 J=JN+1,JR
        JJ=JJ+1
        XX(JJ)=X(IWK(JN,J),J)
        YY(JJ)=Y(IWK(JN,J),J)
  109 CONTINUE
      CALL CURVE(XX,YY,JJ,0)
  110 CONTINUE

C..PLOT NORMAL LINES

      DO 120 I=2,IM/4
      DO 119 J=1,JR
        XX(J)=X(INOR(I,J),J)
        YY(J)=Y(INOR(I,J),J)
  119 CONTINUE
      CALL CURVE(XX,YY,JR,0)
  120 CONTINUE
      CALL ENDPL(0)
      RETURN
      END
      SUBROUTINE AXES(JMAX,OFFSET,LABEL)
C
C    ****************************************************
C    DETERMINE THE LAYOUT OF THE PAGE
C
C    CALLS      : NONE
C
C    CALLED BY : PLOT1, ZONES, WAKE
C    ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)

      COMMON/GRD/X(IDIM,JP1),Y(IDIM,JP1)
```

```
      COMMON/ZN/AL,JVL,JBL,JFL,IV1,IB1,IB2,IV2,ILINE(6),SCALE
      COMMON/PARAM/IM,JR
      CHARACTER*20 LABEL

C..PAGE LAYOUT

      PAGEX=8.25
      PAGEY=10.75
      XAXIS=7.25
      YAXIS=7.25
      CALL BLOWUP(SCALE)
      CALL PAGE(PAGEX,PAGEY)
      CALL AREA2D(XAXIS,YAXIS)
      CALL HEADIN(LABEL,100,1.25,1)

C..COMPUTE GRID AXIS LENGTHS AND SCALES BASED ON
C  MAXIMUM VALUES TO BE PLOTTED

      GRIDMAX=0.
      DO 110 I=1,IM
      DO 109 J=1,JMAX
        IF(ABS(X(I,J)) .GT. GRIDMAX) GRIDMAX=ABS(X(I,J))
        IF(ABS(Y(I,J)) .GT. GRIDMAX) GRIDMAX=ABS(Y(I,J))
  109 CONTINUE
  110 CONTINUE

C..LOCATE THE ORIGIN OF THE PLOT AND DRAW AXES

      XORIG=-GRIDMAX+OFFSET
      YORIG=-GRIDMAX
      XMAX=GRIDMAX+OFFSET
      YMAX=GRIDMAX
      XSTP=1.
      YSTP=1.
      CALL GRAF(XORIG,XSTP,XMAX,YORIG,YSTP,YMAX)
      RETURN
      END
      SUBROUTINE GRID(IMAX1,IMAX2,JMAX)
C
C  ****************************************************
C    GRID IN PHYSICAL PLANE
C
C    CALLS     : NONE
C
C    CALLED BY : PLOT1, ZONES, WAKE
C  ****************************************************
C
      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)

      COMMON/GRD/X(IDIM,JP1),Y(IDIM,JP1)
      COMMON/PARAM/IM,JR
      DIMENSION XX(IP1),YY(IP1)

C..PLOT THE RADIAL LINES

      DO 110 I=1,IMAX1
      DO 109 J=1,JMAX
        XX(J)=X(I,J)
        YY(J)=Y(I,J)
  109 CONTINUE
      CALL CURVE(XX,YY,JMAX,0)
  110 CONTINUE
```

```
C..PLOT AZIMUTHAL LINES
      II=IMAX2
      DO 120 J=1,JMAX
      DO 119 I=1,IMAX2
        XX(I)=X(I,J)
        YY(I)=Y(I,J)
  119 CONTINUE
      IF(IMAX2.EQ.IM) THEN
        XX(IMAX2+1)=XX(1)
        YY(IMAX2+1)=YY(1)
        II=IMAX2+1
      END IF
      CALL CURVE(XX,YY,II,0)
  120 CONTINUE
      RETURN
      END
```

```
      PROGRAM PLOT2

C  ************************************************************
C    THIS PROGRAM ACCEPTS 2D-ARRAYS TO MAKE CONTOURS
C    LAST REVISION 5-25-86
C
C    PRINCIPAL INVESTIGATOR : DR. J.C. WU
C    AUTHOR   : MIKE PATTERSON
C               GEORGIA INSTITUTE OF TECHNOLOGY
C               (404) 894-3028
C
C    TAPE5    : GENERAL INPUT
C    TAPE6    : GENERAL OUTPUT
C    TAPE9    : INPUT FROM ZONST FOR STREAMLINES AND CONTOURS
C    TAPE12   : OUTPUT FOR PLOTTER
C    TAPE14   : INPUT FROM GEOM
C
C    CALLS    : CONTG
C  ************************************************************

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)

      REAL XX(IP1,JP1),YY(IP1,JP1),S1(IP1,JP1),S2(IP1,JP1)
      REAL XB(IDIM),YB(IDIM),CONT(100)
      REAL XB1(IP1),YB1(IP1)
      INTEGER IBUF(512)
      COMMON/SUM/CSQ,GAMA,SIGMA,COSA,SINA
      DATA IRES/2/
      DATA PI/3.14159/

C      REWIND 5
C      REWIND 6
C      REWIND 9
C      REWIND 14

       CALL DIP( 15, 'PLOT2.DIP', 9 )

C..GENERAL INPUT

      READ(5,*) VALMIN,VALMAX,NCON
      READ(5,*) VORMIN,VORMAX,NVOR
      READ(5,*) AHI,ICHR,NDIG,NSKIP,LINTP,HEAVY
      READ(5,*) X1,X2,DX;XLEN
      READ(5,*) Y1,Y2,DY,YLEN
      READ(5,*) IDEV,IOPT,IPRINT,IPAR
      READ(5,*) JMAX,SCALE,NTS,NTL
      IF(NCON.GT.0) READ(5,*) (CONT(I),I=1,NCON)

      IM=IDIM
      IM1=IP1

C..INPUT FROM GEOM

      READ(14) CSQ,GAMA,SIGMA,AL
      READ(14) XB,YB,XX,YY

C..SET DEVICE

C       IF(IDEV.EQ.0) THEN
C         CALL EPIC(300.,0,8.25,10.75,12)
C       ELSE IF(IDEV.EQ.1) THEN
C        CALL PCLCMP
C        CALL CALCMP(IBUF,512,12)
```

```
C      ELSE IF(IDEV.EQ.2) THEN
C       CALL PVRSTC
C       CALL VRSTEC(IBUF,512,12)
C      ELSE IF(IDEV.EQ.3) THEN
C       CALL P4115
C       CALL TK4115(IRES)
C      ENDIF
      CALL SETDEV(6,6)

C..INPUT FROM ZONST — OUTER LOOP — NEW PAGE OF PLOTS

 1000 CONTINUE
      READ(9,END=3000) NT,NPL,T,ALPD,RF,RE,S2,S1
      IF(NT.LT.NTS) GO TO 1000
      IF(NT.GT.NTL) GO TO 3000
      WRITE(6,10) NT,NPL,T,ALPD,RF,RE
      IF(JMAX.GT.NPL) THEN
        WRITE(6,*) 'JMAX > NPL ——EXPECT GARBAGE——'
        STOP
      ENDIF
      ALP=ALPD*PI/180.
      SINA=SIN(ALP)
      COSA=COS(ALP)
      ISUB=0
      INAME=IOPT
      CALL NOBRDR
      CALL NOCHEK
      CALL GRACE(0.)

C..INNER LOOP — SUBPLOTS

 2000 CONTINUE
      ISUB=ISUB+1
      IF(ISUB.EQ.1) CALL PHYSOR(1.5,5.50)
      IF(ISUB.EQ.2) CALL OREL(0.,-4.25)
      CALL COMPLX
      CALL BASALF('STAND')
      CALL MIXALF('L/CSTD')
      CALL MX3ALF('L/CGR',1H/)

      CALL BLOWUP(SCALE)
      CALL AREA2D(XLEN,YLEN)
      CALL YAXANG(0.0)
      CALL XTICKS(0)
      CALL YTICKS(0)

      CALL XNONUM
      CALL YNONUM
      CALL GRAF(X1,DX,X2,Y1,DY,Y2)

      CALL THKFRM(.02)
      CALL FRAME

C..PLOT PARAMETERS

      IF(IPAR.EQ.1) THEN
        WB = 1.55
        HB = 1.05
        XBLK = .05
        YBLK = .05
        CALL BLTREC(XBLK,YBLK,WB,HB,0.0,0.015)
        CALL BLKEY(ID)
        CALL BLOFF(ID)
        XM = XBLK +  0.10
```

```
      YM = YBLK + 0.05
      CALL HEIGHT(0.15)
      CALL MESSAG('R(E  =)$',100,XM,YM)
      XR=XM+.5
      CALL REALNO(RE,-1,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.30
      CALL HEIGHT(0.15)
      CALL MESSAG('(RF  =)$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(RF,3,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.55
      CALL HEIGHT(0.15)
      CALL MESSAG('(T   =)$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(T,3,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.80
      CALL HEIGHT(0.15)
      CALL MESSAG('/A)   =$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(ALPD,3,XR,YM)
      CALL BLON(ID)
     ENDIF

C..DRAW BLADE AT ANGLE OF ATTACK

     DO 140 I=1,IM
       XB1(I)=XB(I)*COSA+YB(I)*SINA
       YB1(I)=YB(I)*COSA-XB(I)*SINA
 140 CONTINUE
     XB1(IM1)=XB1(1)
     YB1(IM1)=YB1(1)
     CALL THKCRV(0.010)
     CALL CURVE(XB1,YB1,IM1,0)
     CALL RESET('THKCRV')

C..DISPLAY DATA, CALL CONTOURING ROUTINE

     IF(IPRINT.EQ.1)
    1 WRITE(6,20)((I,J,XX(I,J),YY(I,J),S1(I,J),S2(I,J),
    2            J=1,JMAX,2),I=1,IM,4)

     IF(ISUB.EQ.1) THEN
     CALL CONTG(S1,XX,YY,1,IM1,1,JMAX,VALMIN,VALMAX,AHI,CONT,NCON,
    1          ICHR,NDIG,NSKIP,LINTP,HEAVY)
     ELSE
     CALL CONTG(S2,XX,YY,1,IM1,1,JMAX,VORMIN,VORMAX,AHI,CONT,NVOR,
    1          ICHR,NDIG,NSKIP,LINTP,HEAVY)
     ENDIF

C..WRITE APPROPRIATE ]ITLE

     XM=XLEN-4.0
     YM=YLEN-.25
     IF(INAME.EQ.-2) THEN
       CALL MESSAG('    S(TREAMLINES)$',100,XM,YM)
     ENDIF
     IF(INAME.EQ.-1) THEN
       CALL MESSAG('V(ORTICITY) C(ONTOURS)$',100,XM,YM)
```

```
      ENDIF
      IF(INAME.EQ.0) THEN
        CALL MESSAG('D(ENSITY) C(ONTOURS)$',100,XM,YM)
      ENDIF
      IF(INAME.EQ.1) THEN
        CALL MESSAG('M(ACH) N(UMBER) C(ONTOURS)$',100,XM,YM)
      ENDIF
      IF(INAME.EQ.2) THEN
        CALL MESSAG('V(ELOCITY) M(AGNITUDE) C(ONTOURS)$',100,XM,YM)
      ENDIF
      IF(INAME.EQ.3) THEN
        CALL MESSAG('S(KIN) F(RICTION) C(ONTOURS)$',100,XM,YM)
      ENDIF

C..FINISH SUBPLOTS OR START NEW PAGE

      CALL ENDGR(ISUB)
      INAME=INAME+1
      IF(ISUB.EQ.1) GO TO 2000
      CALL ENDPL(0)
      GO TO 1000

C..PLOTTING FINISHED

 3000 CALL DONEPL
      STOP
   10 FORMAT(' ','NT,NPL,T,ALPD,RF,RE',2I4,4F13.3)
   20 FORMAT(2(5X,2I5,4F10.5))
      END
      SUBROUTINE CONTG(A,XA,YA,IL,IU,JL,JU,VALMIN,VALMAX,
     1   AHI,CONT,NCON,ICHR,NDIG,NSKIP,LINTP,HEAVY)
C
C   *************************************************************
C     THIS SUBROUTINE PLOTS CONTOURS OF MATRIX "A" FOR GENERAL (X,Y)
C     POSITIONS OF THE ELEMENTS
C
C     CALLS     : XFUN
C
C     CALLED BY : PLOT2
C   *************************************************************
C
      PARAMETER (IP1=81,JP1=61)
      DIMENSION A(IP1,JP1),XA(IP1,JP1),YA(IP1,JP1),CONT(100)
      DIMENSION B(4),X(4),Y(4)
      COMMON/COORD/XX(3),YY(3),XX2(3),YY2(3)
      REAL HT(4)
      LOGICAL HVY
      DATA HT/.08,.10,.12,.14/
      DATA PI/3.14159/

      IF(ICHR.GT.0) CALL HEIGHT(HT(ICHR))
      I2=IU-1
      J2=JU-1
      NSKP=NSKIP
      IF(LINTP.EQ.3) CALL CHNDOT
      IF(LINTP.EQ.4) CALL CHNDSH
      IF(LINTP.EQ.2) CALL DASH
      IF(LINTP.EQ.1) CALL DOT
      IF(NSKIP.LE.0) NSKP=10
      HVY=.FALSE.
      IF(VALMIN.LT.VALMAX) THEN
          VALMX=VALMAX
          VALMN=VALMIN
      ELSE
```

```
          VALMN=1E30
          VALMX=-1E30
          DO 170 I=IL,IU
          DO 170 J=JL,JU
          IF (A(I,J).LT.AHI) THEN
              VALMX=AMAX1(A(I,J),VALMX)
              VALMN=AMIN1(A(I,J),VALMN)
          ENDIF
170       CONTINUE
      ENDIF
      N=NCON
      IF(NCON.LT.0) THEN
       N = -NCON
       CONMIN = VALMIN
       CONINC = (VALMAX - VALMIN)/FLOAT(N-1)
      ENDIF

C..DO LOOP FOR CONTOUR LINES

      DO 500 NCN=1,N
      IF(NCON.LT.0) THEN
        CONTUR=CONMIN+(NCN-1)*CONINC
      ELSE
        CONTUR=CONT(NCN)
      ENDIF
      IF(HEAVY.NE.0.)
     1 HVY=ABS(AINT(ABS(CONTUR/HEAVY)+0.5)-ABS(CONTUR/HEAVY)).LT.0.001
      IF(HVY) CALL THKCRV(.01)
      NSK=NSKP

C..SEARCH MATRIX "A" FOR CONTOURS

      DO 490 I=IL,I2
      DO 490 J=JL,J2
      BB=AMAX1(A(I,J),A(I,J+1),A(I+1,J),A(I+1,J+1))
      IF (BB-AHI) 290,490,490
  290 IF (CONTUR-BB) 300,490,490
  300 IF (CONTUR-AMIN1(A(I,J),A(I,J+1),A(I+1,J),A(I+1,J+1))) 490,310,310
  310 B(1)=.25*(A(I,J)+A(I+1,J)+A(I,J+1)+A(I+1,J+1))
      B(4)=B(1)
      X(1)=.25*(XA(I,J)+XA(I+1,J)+XA(I,J+1)+XA(I+1,J+1))
      X(4)=X(1)
      Y(1)=.25*(YA(I,J)+YA(I+1,J)+YA(I,J+1)+YA(I+1,J+1))
      Y(4)=Y(1)

C..SEARCH MATRIX SUB-CELL TRIANGLES FOR CONTOURS

      DO 480 K=1,4
      NP=0
      IF (K.LE.1) THEN
          B(2)=A(I+1,J)
          X(2)=XA(I+1,J)
          Y(2)=YA(I+1,J)
      ELSE
          B(2)=B(3)
          X(2)=X(3)
          Y(2)=Y(3)
      ENDIF
      IM=I+K/3
      JM=J+1-IABS(5-2*K)/3
      B(3)=A(IM,JM)
      X(3)=XA(IM,JM)
      Y(3)=YA(IM,JM)
```

```
C..DETERMINE CONTOUR INTERSECTIONS

      DO 430 M=1,3
      IF (CONTUR-AMIN1(B(M),B(M+1))) 430,380,380
380   IF (CONTUR-AMAX1(B(M),B(M+1))) 390,390,430
390   NP=NP+1
      BB=B(M+1)-B(M)
      IF (ABS(BB).LE.1.E-15) THEN
         D=0.5
      ELSE
         D=(CONTUR-B(M))/BB
      ENDIF
      XX(NP)=X(M)+D*(X(M+1)-X(M))
      YY(NP)=Y(M)+D*(Y(M+1)-Y(M))
430   CONTINUE
      IF(NP.GT.1) THEN
        IF(ICHR.LT.0) THEN
          IF(CONTUR.GT.0.) CALL RESET('DASH')
          IF(CONTUR.LE.0.) CALL DASH
        ENDIF

C..TRANSFORM CONTOUR TO PHYSICAL PLANE AND DRAW

        CALL XFUN(NP)
        CALL CURVE(XX2,YY2,NP,0)

C..LABEL CONTOURS

        IF(ICHR.GT.0) THEN
          NSK=NSK+1
          IF(NSK.GE.NSKP) THEN
            NSK=-1
            XS=.5*(XX2(1)+XX2(2))
            DXS=XX2(2)-XX2(1)
            YS=.5*(YY2(1)+YY2(2))
            DYS=YY2(2)-YY2(1)
            ANG=180.*ATAN2(DXS,DYS)/PI
            CALL ANGLE(ANG)
            CALL REALNO(CONTUR,NDIG,XS,YS)
          ENDIF
        ENDIF
      ENDIF
480   CONTINUE
490   CONTINUE
      IF(HEAVY.NE.0) CALL RESET('THKCRV')
500   CONTINUE
      RETURN
      END
      SUBROUTINE XFUN(NP)
C
C   ***********************************************************
C     COMPUTES CARTISIAN COORDINATES (XX,YY) IN PHYSICAL PLANE
C     OF A SPECIFIC POINT (X,Y) IN COMPUTATIONAL PLANE
C
C     CALLS      : NONE
C
C     CALLED BY  : PLOT2, CONTG
C   ***********************************************************
C
      COMMON/SUM/CSQ,GAMA,SIGMA,COSA,SINA
      COMMON/COORD/X(3),Y(3),XX(3),YY(3)
      COMPLEX Z1,Z

      DO 100 I=1,NP
```

```
      Z1=GAMA+CMPLX(X(I),Y(I))
      Z=Z1+CSQ/Z1+SIGMA
      X1=REAL(Z)
      Y1=AIMAG(Z)
      XX(I)=X1*COSA+Y1*SINA
      YY(I)=Y1*COSA-X1*SINA
  100 CONTINUE
      RETURN
      END
```

```
      PROGRAM PLOT3

C  *****************************************************
C   LOADS PLOTTING PROGRAM
C   LAST REVISION 6-25-87
C
C   PRINCIPAL INVESTIGATOR : DR. J.C. WU
C   AUTHOR  : MIKE PATTERSON
C             GEORGIA INSTITUTE OF TECHNOLOGY
C             (404) 894-3028
C
C   TAPE5   : GENERAL INPUT
C   TAPE6   : GENERAL OUTPUT
C   TAPE10  : INPUT FROM LOADS PROGRAM
C   TAPE13  : OUTPUT TO PLOTTER
C   TAPE14  : INPUT FROM GEOM
C
C   CALLS   : NONE
C  *****************************************************

      PARAMETER (IDIM=80,JDIM=60)
      PARAMETER (IP1=81,JP1=61)
      PARAMETER (IU=40,IL=42,IMAX=2000)

      REAL XB(IDIM),XU(IU),XL(IL)
      REAL CP(IP1),CPU(IU),CPL(IL)
      REAL T(IMAX),CL(IMAX),CD(IMAX),CM(IMAX),ANG(IMAX)
      REAL XEXP(IU), CPEXP(IU,JDIM), CLEXP(IU), CDEXP(IU),
     >     CMEXP(IU), ALPHA(IU)
      INTEGER IBUF(512),NT(IMAX)
      DATA IRES/2/

C      REWIND 5
C      REWIND 6
C      REWIND 10
C      REWIND 14

C..READ INPUT PARAMETERS

      READ(5,*) SCALE,SCALE2
      READ(5,*) X1,X2,DX,XLEN
      READ(5,*) Y1,Y2,DY,YLEN
      READ(5,*) IOPT,IDEV,IPAR,NTS,NTL

C..INPUT FROM GEOM

      READ(14) DUMM1,DUMM2,DUMM3,AL
      READ(14) XB

C..INPUT FROM EXPERIMENTAL RESULTS

C      READ(24,*) NPARTS
C      READ(24,900) ( XEXP(I),I=1,16 )
C      DO J=1,NPARTS
C        READ(24,900) ALPHA(J), CLEXP(J), CDEXP(J), CMEXP(J)
C        READ(24,900) ( CPEXP(I,J),I=1,16 )
C        DO I=1,16
C          CPEXP(I,J) = -CPEXP(I,J)
C        ENDDO
C      ENDDO

C      IEXP = 0

  900 FORMAT( 5( 1X,E14.7 ) )
```

```
C..SET DEVICE

      CALL DIP( 15, 'PLOT3.DIP', 9 )

C       IF(IDEV.EQ.0) THEN
C         CALL EPIC(300.,0,8.25,10.75,13)
C       ELSE IF(IDEV.EQ.1) THEN
C        CALL PCLCMP
C        CALL CALCMP(IBUF,512,13)
C       ELSE IF(IDEV.EQ.2) THEN
C        CALL PVRSTC
C        CALL VRSTEC(IBUF,512,13)
C       ELSE IF(IDEV.EQ.3) THEN
C        CALL P4115
C        CALL TK4115(IRES)
C       ENDIF
      CALL SETDEV(6,6)
      IM=IDIM

C  *********CP PLOT LOOP***********

      IF(IOPT.EQ.0) THEN

C..INPUT FROM LOADS

 1000 CONTINUE
      READ(10,END=2000) NT1,NTPL,T1,ALPD,RF,RE,CL1,CD1,CM1,CP
      IF(NT1.LT.NTS) GO TO 1000
      IF(NT1.GT.NTL) GO TO 2000
C       IF(MOD(NT1,NTPL).NE.0) GO TO 1000
      WRITE(6,10) NT1,T1,ALPD
  10  FORMAT(' ','NT,T,ALPD :',I5,2F8.3)

C       IEXP = IEXP + 1

C..SET UP ALPHA-NUMERICS AND AXES NAMES

      CALL NOBRDR
      CALL COMPLX
      CALL BASALF('STAND')
      CALL MIXALF('L/CSTD')
      CALL MX3ALF('L/CGR',1H\)
      CALL BLOWUP(SCALE)

C..LABEL AXES

      CALL XNAME('X/C$',100)
      CALL YNAME('-C(P)$',100)

C..DIVIDE DATA INTO UPPER AND LOWER ARRAYS

      DO 100 I=1,IU
        XU(I)=(XB(I)+AL/4.)/AL
        CPU(I)=-CP(I)
  100 CONTINUE
C       IF( ITEST.EQ.0 )THEN
C         WRITE(24,*) ( XU(I),I=IU,1,-1 )
C         ITEST=1
C       ENDIF
      WRITE(24,*) ALPD
      WRITE(24,*) ( CPU(I),I=IU,1,-1 )

      IM2=IM/2-1
```

```
      DO 110 I=1,IL-1
        XL(I)=(XB(I+IM2)+AL/4.)/AL
        CPL(I)=-CP(I+IM2)
  110 CONTINUE
      XL(IL)=XU(1)
      CPL(IL)=CPU(1)

C..DEFINE PLOTTING AREA AND HEADING

      CALL PHYSOR(1.75,3.0)
      CALL AREA2D(XLEN,YLEN)
      CALL SCLPIC(SCALE2)
      CALL YAXANG(0.0)
      CALL XTICKS (5)
      CALL YTICKS (5)
      CALL GRAF(X1,DX,X2,Y1,DY,Y2)
      CALL THKFRM(.02)
      CALL FRAME
      CALL RESET('THKFRM')

C..PARAMETERS

      WB = 2.65
      HB = .25
      XBLK = XLEN-3.5
      YBLK = YLEN+0.4
C     CALL BLTREC(XBLK,YBLK,WB,HB,0.0,0.015)
C     CALL BLKEY(ID1)
C     CALL BLOFF(ID1)
      XM = XBLK +  0.10
      YM = YBLK + 0.05
C     CALL HEIGHT(0.15)
C     CALL MESSAG('P(RESSURE) D(ISTRIBUTION)$',100,XM,YM)

      WB = 1.55
      HB = 1.05
      XBLK = XLEN-1.6
      YBLK = YLEN-1.1
      CALL BLTREC(XBLK,YBLK,WB,HB,0.0,0.015)
      CALL BLKEY(ID2)
      CALL BLOFF(ID2)
      XM = XBLK +  0.10
      YM = YBLK + 0.05
      CALL HEIGHT(0.15)
      CALL MESSAG('R(E  =)$',100,XM,YM)
      XR=XM+.5
      CALL REALNO(RE,-1,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.30
      CALL HEIGHT(0.15)
      CALL MESSAG('(RF  =)$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(RF,3,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.55
      CALL HEIGHT(0.15)
      CALL MESSAG('(T   =)$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(T1,3,XR,YM)

      XM = XBLK +  0.1
      YM = YBLK + 0.80
```

```
      CALL HEIGHT(0.15)
      CALL MESSAG('\A)    =$',100,XM,YM)
      XR = XM + 0.6
      CALL REALNO(ALPD,1,XR,YM)
      CALL BLON(ID1)
      CALL BLON(ID2)

C..PLOT THE CP DISTRIBUTION

      CALL THKCRV(.02)
      CALL MARKER(15)
      CALL CURVE(XU,CPU,IU,0)
      CALL MARKER(18)
C       CALL DASH
C       CALL CURVE(XEXP,CPEXP(1,IEXP),16,0)
C       CALL RESET('DASH')
C       CALL CURVE(XL,CPL,IL,1)
      CALL RESET('THKCRV')
C       CALL GRID(2,1)
      CALL ENDPL(0)
      GO TO 1000

 2000 ENDIF

C  ********END OF CP LOOP, BEGIN LOADS LOOP*********

      IF(IOPT.GE.1) THEN

C..INPUT FROM LOADS

      I=1
 2100 READ(10,END=2200) NT1,NTPL,T1,ALPD,RF,RE,CL1,CD1,CM1,CP
      IF(NT1.LT.NTS) GO TO 2100
      IF(NT1.GT.NTL) GO TO 2200
      WRITE(6,11) NT1,T1,ALPD,CL1,CD1,CM1
   11 FORMAT(' ',I5,5F8.3)
      ANG(I)=ALPD
      T(I)=T1
      CL(I)=CL1
      CD(I)=CD1
      CM(I)=-CM1
      I=I+1
      GO TO 2100
 2200 CONTINUE
      IPTS=I-1

C..INNER LOOP -- SUBPLOTS

      IF(IOPT.EQ.1) CALL PHYSOR(3.0,7.75)
      IF(IOPT.EQ.2) CALL OREL(0.,-3.50)
      IF(IOPT.EQ.3) CALL OREL(0.,-3.50)

C..SET UP ALPHA-NUMERICS AND AXES NAMES

      CALL NOBRDR
      CALL COMPLX
      CALL BASALF('STAND')
      CALL MIXALF('L/CSTD')
      CALL MX3ALF('L/CGR',1H\)
      CALL BLOWUP(SCALE)

C..LABEL AND DRAW AXES

      IF(IOPT.EQ.1) THEN
```

```
          CALL YNAME('L(IFT) C(OEFFICIENT)$',100)
          Y1=-.5
          Y2=2.
          DY=.5
        ENDIF
        IF(IOPT.EQ.2) THEN
          CALL YNAME('D(RAG) C(OEFFICIENT)$',100)
          Y1=-.2
          Y2=1.
          DY=.2
        ENDIF
        IF(IOPT.EQ.3) THEN
          CALL XNAME('\A)$',100)
          CALL YNAME('M(OMENT) C(OEFFICIENT)$',100)
          Y1=-.15
          Y2=.15
          DY=.05
        ENDIF

        CALL AREA2D(XLEN,YLEN)
        CALL YAXANG(0.0)
        CALL XTICKS(5)
        CALL YTICKS(5)
        CALL GRAF(X1,DX,X2,Y1,DY,Y2)

C..PARAMETERS

        IF(IPAR.EQ.1 .AND. IOPT.EQ.1) THEN
          WB = 1.55
          HB = 0.55
          XBLK = 0.05
          YBLK = YLEN-0.6
          CALL BLTREC(XBLK,YBLK,WB,HB,0.0,0.015)
          CALL BLKEY(ID3)
          CALL BLOFF(ID3)
          XM = XBLK +  0.10
          YM = YBLK + 0.05
          CALL HEIGHT(0.15)
          CALL MESSAG('R(E  =)$',100,XM,YM)
          XR=XM+.5
          CALL REALNO(RE,-1,XR,YM)

          XM = XBLK +  0.1
          YM = YBLK + 0.30
          CALL HEIGHT(0.15)
          CALL MESSAG('(RF  =)$',100,XM,YM)
          XR = XM + 0.6
          CALL REALNO(RF,3,XR,YM)
          CALL BLON(ID3)
        ENDIF

C..PLOT THE LOADS DISTRIBUTION

        CALL THKCRV(.017)
        IF(IOPT.EQ.1) CALL CURVE(ANG,CL,IPTS,0)
        IF(IOPT.EQ.2) CALL CURVE(ANG,CD,IPTS,0)
        IF(IOPT.EQ.3) CALL CURVE(ANG,CM,IPTS,0)
C         CALL DASH
C         IF(IOPT.EQ.1) CALL CURVE(ALPHA,CLEXP,20,0)
C         IF(IOPT.EQ.2) CALL CURVE(ALPHA,CDEXP,20,0)
C         IF(IOPT.EQ.3) CALL CURVE(ALPHA,CMEXP,20,0)
C         CALL RESET('DASH')
        CALL RESET('THKCRV')
        CALL THKCRV(.007)
```

```
C      CALL GRID(1,1)
      CALL RESET('THKCRV')
      CALL THKCRV(.0001)
C      CALL GRID(2,2)
      CALL RESET('THKCRV')
      CALL THKFRM(.023)
      CALL FRAME
      CALL RESET('THKFRM')

C..FINISH SUBPLOTS

      CALL ENDGR(0)
      IOPT=IOPT+1
      IF(IOPT.LE.3) GO TO 2200
      CALL ENDPL(0)
      ENDIF

C  ***********END OF LOADS LOOP********

C..PLOTTING FINISHED

 3000 CALL DONEPL
      STOP
      END
```

```
C+-----------------------------------------------------------------
C
      PROGRAM CARPET
C
C  PURPOSE: Creates a single plot with multiple Cp vs  X/C curves in a
C           carpet plot format. The Cp and X/C scales are marked on the
C           first curve drawn. The other curves are drawn to the same
C           scale, but axes are not drawn although the Cp=0.0 level is
C           shown with a dotted line.
C
C
C  EXTERNAL REFERENCES:
C  MODULE      DESCRIPTION
C  DIP         Initializes file in NASA-Ames Device Independent Plot format.
C  DISSPLA     Graphics software.
C  OPENER      Prompts for the name of a sequential file and opens it.(PROGTOOLS)
C  PLFREE      Writes "free values" with headings on a plot          (GRAPHLIB)
C  PLTYTL      Writes a centered character string on the plot.       (GRAPHLIB)
C
C  DEVELOPMENT HISTORY:
C    DATE     INITIALS    DESCRIPTION
C   Nov.1986    RCL       Original design and implementation.
C
C  AUTHOR:     Rosalie C. Lefkowitz  Sterling Software, Palo Alto, CA
C
C-----------------------------------------------------------------

C       IMPLICIT NONE

C  *  Parameter constants:

      INTEGER
     >   LUNCRT, LUNDAT, LUNDIP, LUNIN, LUNKBD, MAXI, MAXI2, MAXJ,
     >   MXSTR, MAXTIT, NFREE
      PARAMETER
     >  ( LUNCRT = 6,
     >    LUNDAT = 8,
     >    LUNDIP = 10,
     >    LUNIN  = 12,
     >    LUNKBD = 5,
     >    MAXI   = 40,
     >    MAXI2  = 40,    !MAXI/2 + 1,
     >    MAXJ   = 20,
     >    MXSTR  = 81,
     >    NFREE  = 3 )


C  *  Variables:

      REAL
     >   ALPHA(MAXJ), CPTH(MAXI,MAXJ), FREVAL(NFREE), XTH(MAXI),
     >   X2(3), Y2(3)

      DIMENSION WORK(MAXI)

      REAL
     >   FRELEN, HIGH, HITFRE, HITITL, WIDE,
     >   XOFSET, XOR, YOFSET, YOR, YINCH, YFREE, YTITLE,
     >   XMAX, XMIN, XSTEP,
     >   CPMAX, CPMIN, CPSTEP

      CHARACTER
     >   DATFIL*80,
     >   FRETXT(3)*(10),
```

```
     >   TITLE*(MXSTR)

      INTEGER
     >   I, J, NDGFRE(NFREE)

      DATA
     >   CPMIN,CPSTEP,CPMAX/ 0.,-20.,-20./,
     >   DATFIL / 'CARPET.DAT' /,
     >   FRELEN/10.0/,
     >   HIGH/3.0/,
     >   HITFRE/0.09/,
     >   HITITL/0.14/,
     >   NDGFRE/2, 2, 3/,
     >   WIDE/2.8/,
     >   XMIN,XSTEP,XMAX/0.,1.,1./,
     >   X2/0.,0.,1./,
     >   XOFSET/.095/,
     >   XOR/0.0/,
     >   YFREE/0.5/,
     >   YOFSET/.16/
     >   YOR/0.0/,
     >   YTITLE/6.5/

C  *  Free text for plot labeling and label for alpha value list:

      WRITE( FRETXT(1), 1100 ) 'AMPLITUDE'
      WRITE( FRETXT(2), 1100 ) 'ST. MEAN'
      WRITE( FRETXT(3), 1100 ) 'RED. FREQ.'

C  *  Initialize plotting device:

      CALL DIP ( LUNDIP, 'CARPET.DIP', 100 )


C  *  Prompt for and open data file:

      CALL OPENER ( LUNCRT,
     >            'Enter data file name (default is CARPET.DAT): ',
     >             LUNKBD, DATFIL, LUNDAT, 'OLD' )

C  *  Read the data file:

      READ (LUNDAT,1000) TITLE
      WRITE(LUNCRT,1000) TITLE
      READ (LUNDAT,*) ( XTH(I),I=1,MAXI )
      READ (LUNDAT,*) ( FREVAL(I),I=1,NFREE )
      DO 10 J=1,MAXJ
         READ (LUNDAT,*) ALPHA(J)
         READ (LUNDAT,*) ( CPTH(I,J),I=1,MAXI )
   10 CONTINUE


      CALL RESET ('ALL')
      CALL SETDEV ( 0, 0 )
      CALL HWROT ('MOVIE')
      CALL HWSCAL ('NONE')
      CALL NOBRDR
      CALL GRACE ( 0.0 )
      CALL NOCHEK
      CALL PAGE ( 11.0, 8.5 )
      CALL BASALF ( 'STANDARD' )
      CALL MIXALF ( 'L/CGREEK')
```

```
C  *  Start with title and free text/free values:

      CALL PHYSOR ( XOR, YOR )
      CALL AREA2D ( 10.0, 8.0 )
      CALL PLTYTL ( 0.0, FRELEN, YTITLE, HITITL, TITLE )
      CALL PLFREE ( NFREE, FRETXT, FREVAL, NDGFRE, HITFRE, YFREE,
     >              FRELEN )
      CALL ENDGR(0)


C  *  Then draw Cp curves:

      XOR = 2.0 - XOFSET
      YOR = 1.0 - YOFSET
      DO 500 J =1,MAXJ

         XOR = XOR + XOFSET
         YOR = YOR + YOFSET
         CALL PHYSOR ( XOR, YOR )
         CALL AREA2D (WIDE, HIGH)
         CALL CROSS
         IF ( J.EQ.1 ) THEN
C  *        First plot has full labeled axes:
            CALL XNAME ( 'X/C', 3 )
            CALL YNAME ( '-Cp', 3 )
         ELSE
C  *        Later curves are drawn without labeled axes:
            CALL XNAME (' ', 0 )
            CALL YNAME (' ', 0 )
         END IF
         CALL GRAF (XMIN, XSTEP, XMAX, CPMIN, CPSTEP, CPMAX)
         CALL CURVE ( XTH(1), CPTH(1,J), MAXI2, 0 )
         IF ( J.GT. 1) THEN
C  *        Draw dotted pseudo-axes:
            Y2(1) = CPTH(1,J)
            Y2(2) = 0.0
            Y2(3) = 0.0
            CALL DOT
            CALL CURVE ( X2, Y2, 3, 0 )
            CALL RESET ( 'DOT' )
         END IF
         CALL HEIGHT ( HITFRE )
C  *     How many inches from the origin is Y= 0 ?
         YINCH = YPOSN ( 0.0, 0.0 )
C  *     Now print the value of ALPHA:
         CALL REALNO ( ALPHA(J), 2, WIDE + 0.2, YINCH )
C  *     Identify the alpha value list:
         IF (J.EQ.MAXJ )
     >   CALL MESSAG ('(a) deg.$', 100, WIDE + 0.2, YINCH + 0.2 )
         CALL ENDGR (0)

  500 CONTINUE

C  *  Termination:

      CALL ENDPL ( 0 )
      CALL DONEPL
      STOP '          Normal termination. Plot file is in CARPET.DIP.'
 1000 FORMAT (A)
 1100 FORMAT (A10)

      END
```

APPENDIX C

NOTES ON THE EMPLOYMENT OF THE WU CODE AT NASA-AMES

DISCLAIMER

The following is a site-specific, step-by-step example intended for the novice user to employ the Wu code at the NASA-AMES computational facility. It makes no attempt to be all encompassing, but rather tries to provide an adequate amount of information to shorten the learning period required to gain a minimum working knowledge of the facilities. Much of it can be applied to the use of other codes at AMES, but it will have only limited application at other sites.

There are a number of useful publications which can be helpful in utilizing the NASA-AMES computational facilities. Included are:

1. INTRODUCTION TO VAX/VMS AT AMES
2. CrayVAX DECnet Interface User's Guide
3. A GUIDE TO GENERATING MOVIES USING PLOT3D AND GAS
4. PLOT3D and PLOT3X Version 3.5 (3D for VAX and 3X for Cray)

Joan Thompson and Rosealie Lefkowitz usually have these publications. Their office is in Bldg 227, Rm 102. Both ladies are very helpful and generous with their time.

You will need access to a VAX account and required password. See an NPS representative for this information. 'Jianps' is an account on RALph currently available for NPS use, which is convenient as DISSPLA and PLOT3D are installed on RALph. Individual VAX are also referred to as 'nodes'.

The following description assumes the user is at a graphics-capable terminal (or 'green screen'). Use for standard terminals will be identical with the exception of plotting locations available. Remote use does not allow for full screen editing.

First, logon by typing 'c ***' , where 'c'is for connect, and '***', is for the first three letters of the node you are logging onto. You will then be prompted for your password (pw). Upper or lower case is allowable for almost all commands.

All regular keyboard entries are submitted to the VAX via the carriage return <cr>, with the exception of command line entries which submitted via the 'Enter' key.

Now that you are logged into the system, you can check to see what you have immediate access to by typing 'dir'.

If the file needed is in a subdirectory, type 'sd [.***]', where '***' is the subdirectory name.

Type 'dir' (or 'list' for added file information) again.

To edit or to look at the file, type 'edt fn' ,where fn is the filename. (i.e. this.file;1 or my.dat) The default version (or number after the semicolon) is the highest number.

You are now in the line editing mode. To go to full screen or keypad editing, type'c' and <cr>.
The keypad is the at the right end of the keyboard. See the "Introduction to VAX/VMS at AMES" for more information. AMES will eventually be converting to UNIX (which, like "VAX" is a unique operating system) and have some different functions and options than are on "DEC". DEC and VAX are normally used

interchangably, but DEC is properly the name of the manufacturer,while VAX is the operating system itself.

After you are finished changing or looking at the file, enter 'PF1' and then keypad '7'. This will put you in the command line mode. To save any changes and create a new version of the file, type 'exit' enter. If no changes have been made or you don't want to save the changes made, type 'quit' and 'Enter'.

To submit a job to the CRAY X/MP-48, type 'csub' and the fn. You will then be prompted for your pw. For additional information, see "Cray VAX DECnet Interface User's Guide".

The following is an example of a series of Cray and VAX runs to employ the Wu code and plotting options.

NOTES WILL BE INSERTED THROUGHOUT.

NOTES : FOR CRAY JOB CONTROL LANGUAGE (JCL), AN '*.' WILL COMMENT OUT A LINE.
ALL CRAY JCL LINES MUST END WITH A PERIOD.
THE FIRST LINE OF THE JCL MUST BE THE 'JOB=' LINE. DON'T HAVE A BLANK LINE FOR THE FIRST LINE OR THE FIRST SPACE ON THE FIRST LINE! IT WON'T WORK.
INSERT YOUR OWN JOB NAME (JN), USER ID (US) AND USER PASSWORD (UPW).
YOUR JN MAY BE REPEATED FOR THE NAME AND VAX ADDRESS (VAX OR NODE ADDRESS FOR THIS CASE IS FML. OTHERS INCLUDE RALph, MARs, TOM, etc.)
'X=' IS FOR YOUR AMES PHONE EXTENSION.
AS OF 19APRIL88, THE FIRST TWO TIME LIMIT SIZES ARE T=150 (CPU SECONDS) AND T=1800. THE FIRST WILL BE ADEQUATE FOR ALL JOBS UP TO ABOUT 100 TIME STEPS. THE CPU LENGTH OF EACH TIME STEP IS DIRECTLY PRORTIONAL THE SIZE OF THE INVISCID REGION BEING CALCULATED. THEREFORE, PRE-STALL ANGLE OF ATTACK CALCULATIONS WILL BE CLOSER TO .5 CPU SECONDS PER TIME STEP AND FULLY STALLED CONDITIONS WILL REQUIRE 1 TO 1.3 CPU SECONDS PER TIME STEP.

THE FOLLOWING IS A POSSIBLE SERIES OF COMPUTER RUNS.

RUN 1: SAVE DATA FOR STEADY STATE CASE

```
JOB,JN=*****,T=150.
ACCOUNT,AC=*****,US=*****,UPW=*****.NAME=***** X=4269 FML::*****
*.
ACCESS,DN=SENDVAX,ID=STTRDM,OWN=RFTRDM.
*.
*. Compile, load and run
*.
*.CFT,ON=CSTAX.(USE THESE OPTIONS FOR ADDITIONAL DEBUGGING INFORMATION)
CFT,ON=A,OFF=CST.(USE THESE OPTIONS FOR ALL REGULAR RUNS.  THEY PROVIDE A
LDR.                 MINIMUM OF EXTRA OUTPUT INFORMATION.)
*.
```

NOTE: ENTER YOUR DESIRED DIRECTORY FOR THE OUTPUT DATA TO BE SENT TO. SENDING THEM TO SCRATCH WILL HELP AVOID EXCEEDING YOUR ALOTTED DISC QUOTA SPACE. YOUR JN CAN BE USED FOR THE *** BELOW.

```
SENDVAX,DN=FT01,VDN='DUA1:[SCRATCH.***]TAPE1.DAT'.    (UNCOMMENT THESE
SENDVAX,DN=FT03,VDN='DUA1:[SCRATCH.***]TAPE3.DAT'.     LINES WHEN USING
SENDVAX,DN=FT14,VDN='DUA1:[SCRATCH.***]TAPE14.DAT'.    THE DISSPLA ROUTINES)
```

NOTES: DISSPLA ROUTINES INCLUDE PLOT1, PLOT2 AND PLOT3.
UNCOMMENT THE FOLLOWING LINE WHEN GENERATING DATA FOR PLOT3D.

```
SENDVAX,DN=FT10,VDN='FAR0:[.NAME]GRID.DAT'.
*.
REWIND,DN=FT02.
SKIPF,DN=$IN.(BYPASSES ANY SPURIOUS LINES AFTER DATA.)
*.
```

NOTE: THE PDN IS FOR THE DATA SAVED FOR EITHER THE INITIAL STEADY STATE CASE (WHICH IS NEEDED FOR ALL DYNAMIC CASES) OR FOR RESTARTS.

```
ACCESS,DN=FT07,PDN=TEST1.
*.
```

NOTE: THIS IS THE START OF THE JCL FOR THE SECOND JOB TO BE RUN. SUBMITTING MULTIPLE JOBS IS IN THIS MANNER IS CALLED 'JOB CHAINING.'

```
*.CFT,ON=CSTAX.
CFT,ON=A,OFF=CST.
LDR.
*.
SAVE,DN=FT08,PDN=TEST1.(USE ONLY WHEN THE DATA GENERATED WILL)
*.                      (BE NEEDED FOR A SUBSEQUENT RUN.)
SENDVAX,DN=FT09,VDN='DUA1:[SCRATCH.***]TAPE9.DAT'.
SENDVAX,DN=FT04,VDN='DUA1:[SCRATCH.***]TAPE4.DAT'.
SENDVAX,DN=FT11,VDN='[.***]Q.DAT'.
/EOF

    0 .15  5.0 1  (ICST=0 TO COMPUTE THE STEADY STATE CASE.)
    .0001 .0005 .0005 75 100
    .4 .3 .6
    10 35 50 74 45 40
    .10 .10  25
     25  25  25   (USE THESE VALUES TO ALLOW A CHECK ON THE OUTPUT SOLUTION.)
```

```
    ICST,RF,ALPS,ICTUR
    WMIN,DFMX,DRMX,KMAX,NCC
    URBI,URBP,URR
    IV1,IB1,IB2,IV2,NPL,NLB
    DTI,DTINC,NTMAX
    NTPL,NTOUT,NTLO
```

RUN 2: A SMALL NUMBER OF TIME STEPS TO CONFIRM THE RESULTS ARE AS DESIRED.

NOTE: ALL THE ABOVE JCL IS THE SAME EXCEPT:

```
*.SAVE,DN=FT08,PDN=TEST1.(USE ONLY WHEN THE DATA GENERATED WILL)
*.                       (BE NEEDED FOR A SUBSEQUENT RUN.)
AND
*.SENDVAX,DN=FT10,VDN='FAR0:[.NAME]GRID.DAT'.

    4 .15  5.0 1  (ICST=1,2,3 OR 4 AS DESIRED.)
    .0001 .0005 .0005 75 100
    .4 .3  6
    10 35 50 74 45 40
    .10 .10  25
     25  25  25   (USE THESE VALUES TO ALLOW A CHECK ON THE OUTPUT SOLUTION.)
```

```
    ICST,RF,ALPS,ICTUR
    WMIN,DFMX,DRMX,KMAX,NCC
    URBI,URBP,URR
    IV1,IB1,IB2,IV2,NPL,NLB
    DTI,DTINC,NTMAX
    NTPL,NTOUT,NTLO
```

RUN 3: THE FINAL DATA CAN BE SAVED ON TAPE IF A RESTART IS TO BE USED.

NOTE: ALL THE ABOVE JCL IS THE SAME EXCEPT

```
SAVE,DN=FT08,PDN=TEST1.(USE ONLY WHEN THE DATA GENERATED WILL)
*.                      (BE NEEDED FOR A SUBSEQUENT RUN.)

    4 .15  5.0 1  (ICST=1,2,3 OR 4 AS DESIRED.)
    .0001 .0005 .0005 75 100
    .4 .3 .6
    10 35 50 74 45 40
    .10 .10  500
     50  50  50   (SELECT OUTPUT TIME STEP VALUES AS DESIRED. THESE VALUES
                   WILL GIVE OUTPUT AND PLOTTING DATA IN THE SAME 4.0 SECOND
                   INTERVALS AS THE MANUAL DOES.  ALTERNATIVELY, ENTER THE
                   SPECIFIC ALLP (OR ALPHA IN DEGREES) VALUES IN ZONST.
                   COMMENT OR UNCOMMENT THE LINES ASSOCIATED WITH ALLP
                   DEPENDING ON THE MANNER OF CHOOSING THE OUTPUT CONTROL.)
    ______________________________

    ICST,RF,ALPS,ICTUR
    WMIN,DFMX,DRMX,KMAX,NCC
    URBI,URBP,URR
    IV1,IB1,IB2,IV2,NPL,NLB
    DTI,DTINC,NTMAX
    NTPL,NTOUT,NTLO
```

AFTER EACH RUN, LOOK AT THE OUTPUT FILE TO SEE IF THE JOB HAS RUN TO COMPLETION. THIS IS INDICATED IF THE CPU TIME IS REASONABLE FOR THE NUMBER OF TIME STEPS RUN, IF THERE ARE NO MESSAGES THAT THE JOB WAS ABORTED, AND THAT NORMAL DATA TRANSFER WAS ACCOMPLISHED. THEN, IF IT HAS, PLOTTING CAN BE DONE.

IF THE WU PLOTTING ROUTINES (DISSPLA) ARE TO BE USED ENSURE, THE FOLLOWING *ARE AVAILABLE IN YOUR DIRECTORY:*

FOR PLOT1:

1. PLOT1.EXE
2. PLOT1.DAT
3. PLOT1.COM

IF YOU DON'T HAVE A VERSION OF PLOT1.EXE IN YOUR DIRECTORY, COMPLETE THE NEXT TWO STEPS.

COMPILE PLOT1.FOR BY TYPING:
    FOR PLOT1   (.FOR AND HIGHEST VERSION
    NUMBER ARE THE DEFAULT.)   THIS WILL CREATE PLOT1.OBJ

LINK PLOT1.OBJ BY TYPING:
    LINK PLOT1,SYS$LIBRARY:INTLIB/LIB,DISSPLA/LIB,INT/LIB
    THIS WILL CREATE PLO1.EXE THIS IS THE FILE NEEDED TO
    ACTUALLY DO THE PLOTTING.

EDT PLOT1.COM TO ENSURE THAT THE PROPER NAMES ARE INCLUDED IN THE FILE NAMES. THESE SHOULD BE THE SAME NAMES AS IN YOUR CRAY JCL.

YOU SHOULD ALSO HAVE SAVED THE DATA BY UNCOMMENTING THE APPROPRIATE LINES IN YOUR JCL. SEE ABOVE.

PLOT1.COM CAN BE:

```
$ GRAPHICS
$ IF "''F$MODE()'" .EQS. "BATCH" THEN SET DEFAULT FAR0:[JIANPS.**]
$ DEFINE/USER FOR001 DUA1:[SCRATCH.***]TAPE1.DAT
```

```
$ DEFINE/USER FOR005 PLOT1.DAT
$ RUN PLOT1
```

WHERE *** IS YOUR JN AND ** IS YOUR DIRECTORY

TO CREATE THE PLOTS AVAILABLE FROM PLOT1, TYPE:
@PLOT1

FOR PLOT2:

1. PLOT2.EXE
2. PLOT2.DAT
3. PL2.COM

IF YOU DON'T HAVE PLOT2.EXE FOLLOW THE ABOVE DIRECTIONS FOR PLOT1.

PLOT2.COM CAN BE:

```
$ GRAPHICS
$ IF "''F$MODE()'" .EQS. "BATCH" THEN SET DEFAULT FAR0:[JIANPS.**]
$ DEFINE/USER FOR001 DUA1:[SCRATCH.***]TAPE1.DAT
$ DEFINE/USER FOR003 DUA1:[SCRATCH.***]TAPE3.DAT
$ DEFINE/USER FOR004 DUA1:[SCRATCH.***]TAPE4.DAT
$ DEFINE/USER FOR009 DUA1:[SCRATCH.***]TAPE9.DAT
$ DEFINE/USER FOR014 DUA1:[SCRATCH.***]TAPE14.DAT
$ DEFINE/USER FOR005 PLOT2.DAT
$ RUN PLOT2
$ DIPQMS/DELETE PLOT2.DIP
```

TO CREATE THE PLOTS AVAILABLE FROM PLOT2, TYPE:
@PL2

FOR PLOT3:

1. LOADS.DAT
2. LOADS.EXE
3. LOADS.COM
4. PLOT3.EXE
5. PLOT3.DAT

IF YOU DON'T HAVE PLOT3.EXE OR LOADS.EXE, SEE ABOVE.

LOADS.COM CAN BE:

```
$ GRAPHICS
$ IF "''F$MODE()'" .EQS. "BATCH" THEN SET DEFAULT FAR0:[JIANPS.**.**]
$ DEFINE/USER FOR003 DUA1:[SCRATCH.***]TAPE3.DAT
$ DEFINE/USER FOR004 DUA1:[SCRATCH.***]TAPE4.DAT
$ DEFINE/USER FOR005 LOADS.DAT
$ RUN LOADS
$ ! DEFINE/USER FOR024 NACA.DAT
$ DEFINE/USER FOR010 FOR010.DAT
$ DEFINE/USER FOR014 DUA1:[SCRATCH.***]TAPE14.DAT
$ DEFINE/USER FOR005 PLOT3.DAT
$ RUN PLOT3
```

TO CREATE THE PLOTS AVAILABLE FROM PLOT3, TYPE:
@LOADS

FOR USING PLOT3D, THESE ARE A SET OF POSSIBLE DATA ENTRIES:

```
GRAPHICS
PLOT3D
READ/2D

THE FOLLOWING ARE RESPONSES TO PROMPTS:
GRID
Q
SUBSETS
1 80 5

1 60 5

1


MINMAX -5 5 -5 5
FU 200
WALLS
1 80

1

1


VECTORS

Y
.1
LINES
Y
.5



P/2D/TEK
```

THIS WILL GIVE A REPRESENTATIVE PLOT OF THE VELOCITY VECTORS. BY CHANGING THE MINMAX AND SUBSETS VALUES, DIFFERENT ASPECTS OF THE FLOW CAN BE HIGHLIGHTED.

THE LAST COMMAND WILL GENERATE A PLOT ON THE TERMINAL. IF A HARD COPY IS DESIRED, AFTER THE PLOT IS VIEWED OR IF YOU ARE SURE YOU KNOW WHAT THE PLOT WILL LOOK LIKE, TYPE: P/DIP. DO THIS FOR EACH OF THE PLOTS THAT YOU WISH TO SAVE. WHEN FINISHED WITH PLOT3D, EXIT FROM IT AND TYPE: DIPQMS Q. THIS WILL QUEUE YOUR PLOTS TO THE LASER PRINTER ON RALPH. USE THE SAME TECHNIQUE TO PRODUCE HARD COPIES OF THE DISPLAY PLOTS. THERE IS ALSO A BATCH SUBMITTAL FILE FOR PLOT2 AND PLOT3.

LIST OF REFERENCES

1. Carr, L.W., "Dynamic Stall Progress in Analysis and Prediction," American Institute of Aeronautics and Astronautics Paper 85-1769-CP, 19-21 August, 1985.

2. Wu, J.C. and Wang, C.M., "Zonal Solution of Unsteady Viscous Flow Problems," American Institute of Aeronautics and Astronautics Paper 84-1637, 25-27 June, 1984.

3. Wu, J.C., "Theory for Aerodynamic Force and Moment in Viscous Flows," American Institute of Aeronautics and Astronautics Paper 80-0011, 14-16 January, 1980.

4. Wu, J.C., and Thompson, J.F., "Numerical Solutions of Time-Dependent Incompressible Navier-Stokes Equations Using an Integrc-Differential Formulation," Computers and Fluids, Vol. 1, 1973.

5. Patterson, M.T., Wu, J.C. and Wang, C.M., "ZETA--A Manual for a Computer Code that Uses a Zonal Procedure for Evaluating Turbulent and Laminar Flows," Georgia Institute of Technology, June 1987.

6. Kuethe, A.M. and Chow, C.Y., Foundations of Aerodynamics: Bases of Aerodynamic Design, John Wiley & Sons, Inc., 1976.

7. Schlichting, H., Boundary-Layer Theory, McGraw-Hill Book Company, 1979.

8. Shapiro, A.H., The Dynamics and Thermodynamics of Compressible Fluid Flow, Vol. 2, The Ronald Press Company, 1954.

9. Wu, J.C., "Problems of General Viscous Flows," Chapter 4 of Developments in Boundary Element Methods--2, Edited by R. Shaw and P. Banerjee, Applied Science Publishers, 1982.

10. Wu, J.C., "Fundamental Solutions and Numerical Methods for Flow Problems," International Journal for Numerical Methods in Fluids, Vol. 4, John Wiley & Sons, Ltd., 1984.

11. Anderson, D.A., Tannehill, J.C. and Pletcher, R.H., Computational Fluid Mechanics and Heat Transfer, Hemisphere Publishing Corporation, 1984.

12. Wu, J.C. and Guliat, V., "Separate Treatment of Attached and Detached Flow Regions in General Viscous Flows," American Institute of Aeronautics and Astronautics Paper 79-1451, 23-24 July, 1979.

13. Wu, J.C., "Numerical Boundary Conditions for Viscous Flow Problems," American Institute of Aeronautics and Astronautics Paper 75-47, 20-22 January, 1975.

14. McAlister, K.W., Puici, S.L., McCroskey, W.J. and Carr, C.W., "An Experimental Study of Dynamic Stall on Advanced Airfoil Sections," National Aeronautics and Space Administration Technical Memorandum 84245, September 1982.

15. Valdes, J.F., Dynamic Stall Calculations Using a Navier-Stokes Solver, Master's Thesis, Naval Postgraduate School, Monterey, CA, December 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code 67 Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	5
4.	Curricular Officer, Code 31 Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5000	1
5.	Chief Fluid Mechanics Laboratory Mail Stop 260-1 NASA Ames Research Center Moffett Field, California 94035	3
6.	Dr. Lawrence W. Carr MS 260-1 NASA Ames Research Center Moffett Field, California 94035	2
7.	Mr. Greg Howe MS 227-6 NASA Ames Research Center Moffett Field, California 94035	1
8.	Mr. Thomas Momiyama Head, Aircraft Division NAVAIR Code 931 Washington, D.C. 20361	1
9.	Dr. J.C. Wu Department of Aerospace Engineering Georgia Institute of Technology Atlanta, Georgia 30332-0150	1

10. Mr. Mike Patterson 1
Box 31173
Atlanta, Georgia 30332

11. Mr. Harry Berman 1
NASC Headquarters Code 931M
Washington, D.C. 20361

12. LT Jack H. Conroyd, Jr. 5
NTS BOQ
Bldg. 82
San Diego, California 92133-1000

13. Dr. John A. Ekaterinaris 1
NASA Ames Research Center
Mail Stop 258-1
Moffett Field, California 94035

14. Ms. Lisa Cowles 1
NADC Code 60C
Street Rd.
Warminster, Pennsylvania 18974-5000

15. Mr. J.H. Conroyd, Sr. 1
10723 W. 5th Ave.
Countryside, Illinois 60525

END
DATE
FILMED
DTIC
10-88